Il BASIC e il
COMMODORE 64
in pratica
H. Peckham
W. Ellis, Jr. e E. Lodi

Il BASIC e il COMMODORE 64 in pratica

Il BASIC e il COMMODORE 64 in pratica

H. Peckham

W. Ellis, Jr. e E. Lodi

McGRAW-HILL Book Company GmbH

Amburgo · New York · St Louis · San Francisco · Auckland · Bogotá · Città del Guatemala · Città del Messico · Johannesburg · Lisbona · Londra · Madrid · Montreal · Nuova Delhi · Panama · Parigi · San Juan · San Paolo · Singapore · Sydney · Tokyo · Toronto

Un libro di **BYTE**

Titolo originale: *Hands-On BASIC for the Commodore 64*

Realizzazione editoriale: Edigeo snc, via Ozanam 10a, 20129 Milano
Traduzione: Vittoria De Stefani
Grafica di copertina: Valentina Boffa
Composizione e stampa: Litovelox, Trento

ISBN 88-7700-009-0

1ª edizione Settembre 1984
1ª ristampa Gennaio 1985

Indice

Prefazione

Questo libro, specificatamente rivolto al Commodore 64, è una rielaborazione di un precedente e fortunato lavoro, *BASIC: A Hands-on Method*, del quale conserva l'impostazione.
I libri di programmazione BASIC hanno in genere due seri inconvenienti: primo, quasi tutti presuppongono da parte del lettore una buona conoscenza della matematica; secondo, raramente "impongono" all'utente lunghi esercizi al computer. La nostra esperienza ci ha confermato che un principiante si impadronisce di concetti nuovi più facilmente e più rapidamente quando la teoria è preceduta da una buona quantità di esperimenti pratici. Esistono certo corsi di programmazione ma la quasi totalità degli utenti di personal impara a programmare per conto proprio. Questo libro si rivolge a tutti coloro che, da soli o in gruppo, con le conoscenze matematiche più diverse, vogliono imparare a utilizzare il Commodore 64.
La struttura del libro è finalizzata a semplificare l'apprendimento. Ogni capitolo comincia col fissare gli obiettivi; poi gli esercizi di scoperta permettono agli studenti di fare esperimenti col BASIC e vedere il linguaggio in azione. Una volta presa confidenza col C-64, si può passare a trattazioni più tradizionali dei concetti base. I test infine permettono un autocontrollo dell'apprendimento.
Ognuno dei 12 capitoli richiede una o due ore di lavoro al calcolatore e altrettante di studio.
È necessario avere a disposizione un Commodore 64 con video, in bianco e nero o a colori, e un drive. Quest'ultimo non è indispensabile, ma averne uno faciliterà di molto il vostro approccio al computer. Nel testo si farà spesso riferimento alla manualistica fornita con il C-64: la *Guida di ri-*

ferimento per il programmatore e il *VIC-1541 Single Drive Floppy Disk User's Manual.*

RINGRAZIAMENTI

Desidero ringraziare Wade Ellis Jr. e Ed Lodi della Computer Tutors di San Josè, California, per la loro assistenza nell'adattamento di *BASIC: A Hands-on Method* e per la composizione di questo libro. Un grazie anche ad Antonio Padial e Loralee Windsor per la consulenza editoriale prestata.

Herbert D. Peckham

Desideriamo ringraziare Jane Ellis che ha lavorato al computer per gli esercizi e le soluzioni dei test di apprendimento. Lo staff del Context Project della Stanford University è stato, come sempre, di grande aiuto. Speciali ringraziamenti sono dovuti a Dikran Karagueuzian che si è occupato dei programmi di gestione dei file.

Wade Ellis, Jr. e Ed Lodi

Introduzione

I computer fanno ormai parte della nostra vita. Possiamo non accorgercene, ma essi esistono, impiegati nella maggior parte delle nostre attività quotidiane. Industrie di varie dimensioni, istituzioni accademiche, amministrazioni pubbliche: senza i computer nessuna di queste sarebbe in grado di manipolare l'incredibile quantità di informazioni che caratterizza la nostra società. Solo negli ultimi anni l'uso del computer è divenuto una parte significativa delle attività quotidiane e questa tendenza continuerà di sicuro: un sempre maggior numero di persone avrà bisogno di sapere come usare i computer se vorrà prender parte attiva nella società.

CHE COS'È IL BASIC?

State per intraprendere lo studio di un linguaggio per computer chiamato BASIC. Il BASIC è un linguaggio molto specializzato studiato per permettere a voi e al computer di capirvi e comunicare l'un con l'altro. Il BASIC possiede un vocabolario semplice, formato da poche parole, una struttura grammaticale e delle regole d'uso, proprio come ogni altra lingua. Non è complicato ed è certamente più facile da imparare di una lingua parlata come l'inglese o il francese.
L'intendimento di questo libro è quello di insegnarvi il vocabolario del BASIC, farvi familiarizzare con le sue regole grammaticali, e permettervi di usare il computer per fare quello che volete. Il livello della matematica è stato intenzionalmente tenuto molto basso, perciò, se vi sentite un po' arrugginiti nelle vostre capacità in merito, non preoccupatevene troppo. Man mano che andremo avanti con il BASIC, avrete modo di rinfre-

scare alcune cognizioni elementari di matematica.
Un metodo molto efficace per imparare qualcosa è quello di osservare ciò che avviene mentre si esegue un certo compito: il metodo della "scoperta". È la strategia che verrà usata in questo libro. Vi sarà chiesto di cominciare ciascun capitolo con una seduta di lavoro sul computer. Dopo aver seguito le istruzioni ed aver visto quello che il computer fa in risposta alle vostre istruzioni, comincerete ad acquisire una certa "sensibilità" per il BASIC. Quando avrete questo tipo di conoscenza, potrete procedere con maggiore profitto nello studio delle parti teoriche che riassumono quanto avrete imparato. L'esercizio diretto sul computer è quindi in questo libro una parte integrante dell'apprendimento del BASIC.

DOVE È NATO IL BASIC?

La versione originale del BASIC è stata creata al Dartmouth College sotto la direzione dei professori John G. Kemeny e Thomas E. Kurtz. Nel settembre del 1963 essi iniziarono i lavori su un progetto tendente alla creazione di un linguaggio di programmazione scritto dal punto di vista dell'utente. Un aspetto molto interessante della cosa è che gran parte del lavoro effettivo di programmazione sul progetto fu effettuato a Dartmouth da studenti universitari non ancora laureati. La data di nascita ufficiale del BASIC è il 1° maggio 1964.
Il successo di questo sforzo da pionieri compiuto a Dartmouth attirò l'attenzione e ben presto altre istituzioni ne furono interessate. Il resto è storia. Oggi praticamente qualunque computer usa una qualche versione del linguaggio BASIC. Le versioni perfezionate del BASIC hanno aumentato in modo significativo la potenza e le capacità del linguaggio originale.
Lo sviluppo più recente è l'uso del BASIC su computer piccoli ed economici. Il BASIC C-64, il linguaggio presentato in questo libro, è una potente e flessibile versione perfezionata del BASIC.

COME USARE QUESTO LIBRO

Ciascun capitolo inizia con una breve dichiarazione degli obiettivi. Questi devono essere studiati attentamente in modo da ottenere una visione chiara di quello che dev'essere fatto. Segue poi la sezione degli Esercizi di scoperta: quando vi viene richiesto, dovete scrivere la risposta del computer nello spazio previsto nel testo. A volte vi sarà chiesto di rispondere a delle domande. Lo scopo di questa attività è quello di condurvi attraverso i concetti trattati e permettervi di vedere il BASIC al lavoro. È importante che cerchiate di pensare a che cosa succederà in determinate situazioni. Si tratta di una relazione attiva fra voi e il vostro computer e

non dev'essere trascurata. Non è importante che le vostre risposte siano giuste o no; l'importante è che pensiate attentamente alle domande e che cerchiate di rispondere. Il tempo impiegato in questa attività vi farà risparmiare tempo in seguito.
Dopo gli esercizi di scoperta di ciascun capitolo, troverete una completa trattazione di tutti gli obiettivi. Dal momento che avrete già visto le idee e i concetti in azione sul computer, lo studio di questo materiale sarà molto più facile e proficuo.
In ciascun capitolo sono compresi dei programmi completi e funzionanti. Questi sono esaminati in dettaglio per mettere in evidenza come gli elementi della programmazione sono raggruppati per produrre un programma in BASIC.
Ciascun capitolo — a partire dal Capitolo 4 — ha una raccolta di problemi. Cercate di risolverne il più possibile, fino a che vi sentirete in grado di scrivere programmi del livello adeguato a ciascun capitolo. Le soluzioni dei problemi dispari si trovano alla fine del libro.
Infine, ogni capitolo (escluso il primo) ha un test di controllo della preparazione. Lo scopo di questo controllo è quello di provare la vostra comprensione del materiale e di mettere in evidenza gli argomenti che richiedono uno studio più approfondito. Le risposte ai test di controllo della preparazione sono contenute in una sezione alla fine del libro.

Capitolo

Prendere confidenza con il Commodore 64

1

Il vostro primo contatto con il computer può essere un po' faticoso e complicato, perciò andremo avanti molto lentamente. Dopo poche sedute, le operazioni più usuali vi sembreranno molto naturali e non vi procureranno alcuna difficoltà. Siate preparati comunque a una certa confusione all'inizio. Se necessario, non esitate a ripassare il materiale già studiato.

1.1 Obiettivi

In questo capitolo vogliamo prender confidenza con il computer e cominciare ad imparare come funziona. Non verrà effettuata alcuna programmazione BASIC fino al prossimo capitolo. Imparare come funziona la tastiera e come le informazioni vengono immesse e modificate è facile ma fondamentale per tutto ciò che seguirà.

ACCENSIONE DEL COMPUTER

Esamineremo alcune fasi preliminari come l'accendere e spegnere il computer, collegare la TV e selezionare il canale.

MODO DIRETTO

Uno dei modi più facili di usare il computer è quello diretto. Non è richiesta alcuna programmazione: il computer esegue le istruzioni man mano che vengono immesse. A suo tempo impareremo come fare molto di più, ma per ora delle semplici operazioni in modo diretto costituiscono una buona introduzione all'uso del computer.

CORREGGERE GLI ERRORI

È raro che le informazioni siano immesse nel computer senza errori. Si ha quindi bisogno di poter cambiare o correggere facilmente quanto è stato immesso.

1.2 Esercizi di scoperta

Prima di iniziare il lavoro sul computer, dobbiamo fissare alcuni punti molto importanti. Su una macchina da scrivere la L minuscola viene spesso usata al posto del numero 1. Sul computer invece il numero 1 si trova con gli altri tasti numerici nella prima fila della tastiera. Ugualmente, non bisogna usare la lettera O maiuscola o minuscola al posto del numero 0. Come il numero 1, lo 0 si trova nella prima fila di tasti della tastiera.

Non usare la L al posto dell'1
Non usare la O al posto dello 0

Per battere il segno + non occorre premere il tasto SHIFT; lo stesso vale per i simboli − * ↑.
Il drive non sarà usato fino al Capitolo 3: per ora non occorre installarlo né accenderlo.
Il calcolatore deve essere collegato con il televisore. Il selettore dei canali deve essere posizionato su UHF. Istruzioni più dettagliate sul collegamento TV-computer si trovano sul manuale d'uso del C-64.

1. Ora siamo pronti per cominciare a lavorare. Sedetevi di fronte al computer, mettetevi comodi e individuate il tasto RETURN sul lato destro della tastiera.

2. Accendete il televisore; assicuratevi che non siano inserite cartucce nel retro del C-64 e accendete il computer con il tasto ON/OFF posizio-

nato sul lato destro. Una luce rossa vi indicherà che il computer è acceso. Appena la TV si sarà scaldata comparirà la presentazione:

```
    **** COMMODORE 64 BASIC V2 ****

 64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
▒
```

Sintonizzate la TV per una visione più chiara. Se il messaggio d'apertura non compare, ripetete la procedura. Il quadrato lampeggiante è chiamato cursore; sarà quasi sempre presente sul video.

3. Ora battete

   ```
   PRINT 1+4
   ```

   Provate a premere simultaneamente i tasti SHIFT e COMMODORE C= e continuate a battere. (Non premete SHIFT quando dovete scrivere +). È successo qualcosa?

   ....

   Ora premete RETURN e scrivete qui sotto cosa è successo.

   ....

4. Ora sapete come si fa a far fare le addizioni al computer. Vediamolo un po' più a fondo. Battete

   ```
   PRINT 20+54.7
   ```

   e premete RETURN. Che cosa è successo?

   ....

5. Battete

   ```
   PRINT 2+4-3
   ```

   e premete RETURN. Registrate qui sotto il risultato.

   ....

6. Battete ora

```
PRINT 12/2
```

e premete RETURN. Che cosa è successo?

....

Qual è l'operazione aritmetica richiamata dal segno / ?

....

7. Se nel battere dei tasti fate un errore, potete spostare il cursore indietro sull'errore premendo il tasto segnato INST/DEL situato in alto a destra. Citeremo questo tasto chiamandolo tasto DELETE. Ogni volta che il tasto DELETE viene premuto, il cursore si sposterà di uno spazio a sinistra, cancellando un carattere. Quando raggiungete l'errore, ribattete la riga in modo corretto. Quando premete il tasto RETURN, il computer può rispondere con **SYNTAX ERROR** (Errore di sintassi). Se questo accade, cercate di vedere qual è il problema e ribattete la riga.

8. Il vostro video dovrebbe essere abbastanza pieno ora. Premete il tasto SHIFT e quello CLR/HOME, vicino al tasto DELETE. Chiameremo l'operazione: SHIFT/CLR/HOME. Che cosa è successo?

....

Premendo SHIFT/CLR/HOME si cancella lo schermo. Se il cursore non appare sullo schermo, premete RETURN diverse volte. Se si aggiungono linee quando lo schermo è pieno, le precedenti, scorrendo verso l'alto, scompariranno.

9. Battete

```
PRINT 2*50
```

e premete RETURN. Non premete SHIFT quando battete *. Che cosa è successo?

....

Qual è l'operazione aritmetica richiamata da * ?

....

10. Battete la seguente espressione ma non premete il tasto RETURN

```
PRINT (2+3)*4-1
```

Che cosa pensate che succeda quando si preme RETURN?

....

Premete RETURN e scrivete qui sotto quello che è successo.

....

11. Battete ora

```
PRINT "(2+3)*4-1"
```

e premete RETURN. Che cosa ha fatto il computer?

....

12. Che cosa accadrà se battete

```
PRINT "TORO SEDUTO"
```

e premete RETURN?

....

Provate a farlo e vedete se avevate ragione.

13. Passiamo ora ad un argomento diverso. Per prima cosa cancellate lo schermo. Se avete dimenticato come si fa, ripassate il punto 8. Battete la riga seguente. Premete il tasto RETURN quando avete finito.

```
GRADI=95
```

Ora battete

```
PRINT GRADI
```

e premete RETURN. Che cosa è successo?

....

Ora battete

```
?GRADI
```

e premete RETURN. Cosa è successo?

....

Il punto interrogativo è l'abbreviazione di PRINT

14. Esaminate con calma le righe seguenti.

```
LARGHEZZA=10
PROFONDITA'=6
ALTEZZA=4
VOLUME=LARGHEZZA*PROFONDITA'*ALTEZZA
PRINT VOLUME
```

Che cosa pensate che farà il computer se immetterete queste righe?

....

Ora battete le righe ricordandovi di premere RETURN alla fine di ciascuna riga. Che cosa è successo?

....

15. Studiate brevemente le righe seguenti.

```
LARGHEZZA=12
PROFONDITA'=9
METRIQUADRI=(LARGHEZZA*PROFONDITA')
PRINT METRIQUADRI,"METRIQUADRI"
```

Che cosa farà il computer con queste istruzioni?

....

Cancellate lo schermo e battete le righe suddette. Ricordatevi di premere RETURN dopo ogni riga. Che cosa ha fatto il computer?

....

16. Passiamo ora ad un argomento diverso. Premete il tasto SHIFT LOCK,

sulla sinistra della tastiera. Ora premete i tasti A, S, D, F, G, H, J, K e L. Cosa è successo?

....

Premendo SHIFT LOCK si pone la tastiera del computer in modo grafico. Osservate sulla faccia anteriore dei tasti che avete premuto il carattere grafico generato in questo modo. Non tutti i tasti generano caratteri grafici. Provate gli altri tasti.

17. Ora premete SHIFT e il tasto C= . Cosa è successo?

....

Ripetete questa operazione più volte osservandone i risultati sullo schermo.

18. Qui finisce l'attività di scoperta per questo capitolo. Spegnete il computer e il televisore.

1.3 Analisi

ACCENSIONE DEL COMPUTER

Il computer è estremamente semplice da accendere (ON) o spegnere (OFF). Come avete già visto, questo si fa azionando l'interruttore situato nella parte destra del computer. Prima di accendere il computer assicuratevi di aver rimosso l'eventuale cartuccia dal retro della macchina.

Non rimuovete o inserite una cartuccia mentre il C-64 è acceso

Bisogna poi accendere il televisore: comparirà sullo schermo il messaggio di presentazione e

```
READY
▒
```

Il computer all'accensione è in modo maiuscolo/grafico. Si può alternare il modo maiuscolo e minuscolo premendo SHIFT e C= simultaneamente. In questo libro verrà usato sempre il modo maiuscolo/grafico.
Una cosa importante: se in qualunque momento avete l'impressione che le cose vi sfuggano di mano, se avete perso il contatto o se il computer

sembra aver perso il controllo, esiste un meccanismo sicuro per uscirne. Semplicemente, spegnete il computer (OFF). Lo svantaggio di questa operazione è che tutti i programmi e le informazioni in memoria andranno persi.
In alternativa potete premere RUN/STOP e RESTORE: ciò cancella le istruzioni in modo diretto sullo schermo ma non le informazioni già memorizzate.

MODO DIRETTO

Nelle attività di scoperta avete appreso come fare delle semplici operazioni aritmetiche usando il computer come una calcolatrice. Questo è anche conosciuto come modo diretto. Come vedremo nel prossimo capitolo, il BASIC immagazzina istruzioni e comandi in una serie di righe numerate, e quindi è diretto da voi ad eseguire tutte le istruzioni allo stesso tempo. Se, però, le istruzioni vengono battute senza un numero di riga, il computer presume che si voglia una risposta diretta o immediata ed esegue quello che gli è stato chiesto di fare, se questo è possibile.
Quando le istruzioni sono battute, non succede nulla fino a che non si è premuto il tasto RETURN. Il tasto RETURN dice al computer che si è completata l'istruzione. In alcuni casi il computer risponde ad un'unica pressione di tasto e non richiede che venga premuto il tasto RETURN. Questi casi, tuttavia, sono delle eccezioni, piuttosto che la regola.
Abbiamo scoperto che l'addizione e la sottrazione sono richiamate da + e −, il che probabilmente non è stato una grande sorpresa! La moltiplicazione e la divisione sono indicate rispettivamente da * e /. Le parentesi possono essere usate per raggruppare operazioni in qualunque modo si desideri. C'è un certo numero di altre operazioni più sofisticate che possono essere eseguite, ma ne rimandiamo la trattazione ai capitoli successivi. Se si batte

```
PRINT 5*3.2+6.3
```

e si preme RETURN, il computer esegue le operazioni aritmetiche e stampa il risultato.
Se si batte

```
PRINT "ABCDEFG"
```

e si preme RETURN, al computer viene dato l'ordine di stampare la sequenza di caratteri compresa fra le virgolette, in questo caso le lettere ABCDEFG. Tale sequenza è chiamata "stringa di caratteri", ed è un concetto importante sul quale torneremo nel corso del manuale.
Il computer può eseguire diverse istruzioni nel modo diretto. Così

```
A=2
B=3
PRINT A+B
```

farà sì che sullo schermo venga stampato 5. C'è una cosa molto importante collegata con questo concetto. Se battiamo

```
PRINT TASSE
```

e quindi premiamo RETURN verrà visualizzato il numero zero. Dal momento che non abbiamo fornito alcun valore per TASSE, il computer ha assegnato il valore 0 e quindi l'ha stampato.
Il computer è molto elastico per quanto riguarda i nomi per le quantità usati sia nel modo diretto che nei programmi BASIC. Possiamo usare nomi "lunghi" come LUNGHEZZA o TASSE, e nomi "brevi" come L o T. I nomi lunghi possono creare dei problemi: il computer infatti utilizza solo i primi due caratteri per distinguere le variabili e interpreta PENNA e PESCE allo stesso modo. I nomi non devono includere spazi. Alcune parole non possono essere usate come nomi di variabili dal momento che sono riservate per l'uso da parte del computer. L'elenco delle "parole riservate" è contenuto nell'appendice A della *Guida di riferimento per il programmatore*.

CORREGGERE GLI ERRORI

Più avanti impareremo ad usare i comandi per correggere programmi BASIC. Si può tuttavia apportare correzioni in modo diretto, prima di aver premuto RETURN. Potete correggere una riga muovendo il cursore verso sinistra con il tasto DELETE. Ogni volta che si preme il tasto DELETE, si cancella un carattere alla sinistra del cursore; dopo la correzione si può riprendere a battere.

1.4 Test di apprendimento

Effettuate il seguente test per scoprire se avete imparato bene gli obiettivi del Capitolo 1. Le risposte sono date nell'appendice C.

1. Quando avete finito di battere una riga, come fate per farlo sapere al computer?

   ....

2. Se perdete il controllo del computer, come potete riprenderlo?

   ....

3. Qual è il simbolo usato per indicare la moltiplicazione sul computer?

   ....

4. Come si fa a cancellare il video?

   ....

5. Quale operazione indica il simbolo /?

   ....

6. Che cosa succede se battete

   ```
   PRINT 3*4/6
   ```

   e poi premete RETURN?

   ....

7. Che cosa succederà se battete

   ```
   PRINT "25/5+2"
   ```

   e poi premete RETURN?

   ....

8. Supponete di aver battuto PRING 2=3*4 e prima di premere RETURN vi accorgete che al posto della T nella parola PRINT c'è una G. Descrivete come si deve fare per correggere l'errore.

   ....

Capitolo

Introduzione al BASIC 2

Ora siamo pronti per cominciare ad imparare a programmare in BASIC. In questo capitolo vedremo come scrivere ed eseguire alcuni programmi molto semplici.

2.1 Obiettivi

CORREGGERE I PROGRAMMI

Già sapete come correggere una riga usando il tasto DELETE. Vedremo ora come sostituire una riga errata di un programma, battendo una nuova riga con lo stesso numero. Impareremo inoltre ad usare le capacità di correzione, (*editing*), del BASIC C-64 per aggiungere o cancellare informazioni da una riga. Una buona conoscenza dell'editing vi farà risparmiare tempo in seguito.

REQUISITI DEI PROGRAMMI BASIC

Tutti i programmi BASIC hanno delle caratteristiche comuni. Ne osserveremo alcuni molto semplici per imparare quali sono queste caratteristiche.

DIRE AL COMPUTER QUEL CHE DEVE FARE

I comandi sono usati per dire al computer di fare qualcosa ad un programma BASIC o con un programma BASIC. Osserveremo i seguenti comandi: LIST, RUN e NEW.

IMMISSIONE E CONTROLLO DEI PROGRAMMI

Questo obiettivo si sovrappone in parte a quello precedente. La cosa principale che vogliamo raggiungere è di farvi sentire a vostro agio quando immettete e controllate i programmi. Tutti i programmi che incontreremo inizialmente sono brevi e facili da gestire.

NOMI DELLE VARIABILI IN BASIC

Dobbiamo sapere come chiamare i numeri o le stringhe di caratteri nei programmi in BASIC. Fortunatamente il computer ha delle regole molto elastiche in proposito.

2.2 Esercizi di scoperta

Nelle attività di scoperta che seguono, vi verrà suggerito come immettere vari programmi. Se vedete un <RETURN> nelle istruzioni, premete il tasto RETURN. Ricordate dalle vostre esperienze del Capitolo 1 che premendo il tasto RETURN si avverte il computer che si è finito di battere la riga. Ora passate alle attività illustrate qui di seguito.

1. Accendete il computer e il televisore.

2. Battete

   ```
   100 LET A=1 <RETURN>
   ```

   Questa è la prima riga di un programma BASIC.

3. Ora battete il resto del programma come è elencato qui sotto.

   ```
   110 LET B=8 <RETURN>
   120 LET C=A+B <RETURN>
   130 PRINT C <RETURN>
   140 END <RETURN>
   ```

Se fate errori durante la battitura del programma, ribattete la riga o correggete l'errore usando i metodi appresi nel Capitolo 1. Se vi accorgete di aver commesso un errore dopo aver premuto RETURN, battete, in qualsiasi punto del programma, una riga con lo stesso numero, che si sostituirà a quella errata.

4. Cancellate lo schermo usando il comando SHIFT/CLR/HOME. Che cosa è successo al programma che avete appena battuto?

   ....

5. Fortunatamente non tutto è perduto. Il computer ricorda quello che avete battuto anche se lo schermo è diventato vuoto.
   Battete LIST e premete il tasto RETURN. Che cosa è successo?

   ....

6. Sullo schermo dovreste vedere il programma che avete appena immesso. Per ora ignorate i numeri posti all'inizio di ciascuna riga. Leggete le righe del programma e cercate di capire che cosa significano, pensando che LET in inglese significa "poniamo che", PRINT "stampa" e END "fine". Se al computer viene detto di eseguire le istruzioni, che cosa pensate che succederà?

   ....

   Battete RUN ("esegui") e quindi premete il tasto RETURN. Che cosa è successo?

   ....

7. Ora battete

   ```
   110 LET B=5 <RETURN>
   ```

   Cancellate lo schermo, battete LIST, e quindi premete il tasto RETURN. È un 8 o un 5 l'ultimo numero della riga 110?

   ....

8. Se dite al computer di eseguire questo programma che cosa pensate che accadrà?

   ....

Battete RUN, premete il tasto RETURN e registrate qui sotto quello che avviene. Avevate visto giusto?

....

9. Battete ora

```
140 <RETURN>
```

Cancellate lo schermo e visualizzate il programma usando il comando LIST. Che cosa è successo alla riga 140?

....

Se volete cancellare una riga in un programma BASIC, in che modo lo potete fare?

....

10. Ora eseguite il programma battendo RUN e RETURN. Che cosa succede?

....

Vi sembra che l'istruzione END ("fine") che si trovava prima nella riga 140 sia richiesta dal computer?

....

11. Proviamo a fare qualche altra prova. Spesso vorremo cancellare il programma dalla memoria del computer. Questo si ottiene con il comando NEW ("nuovo") che cancella la memoria. Battete NEW e premete il tasto RETURN. Che cosa succede?

....

Battete LIST e premete il tasto RETURN per vedere quello che il computer ha in memoria. C'è qualcosa in memoria?

....

12. Abbiamo imparato come cancellare un programma dalla memoria, ma adesso non abbiamo più il programma! Per riaverlo a disposizione dobbiamo immetterlo di nuovo. Battete il programma seguente.

```
100 LET A=1 <RETURN>
110 LET B=8 <RETURN>
120 LET C=A+B <RETURN>
130 PRINT C <RETURN>
140 END <RETURN>
```

Controllate tutte le righe per assicurarvi che siano state immesse in modo corretto. Se una riga ha bisogno di essere cambiata, ribattetela. Se avete dovuto ribattere delle righe, cancellate lo schermo con il comando SHIFT/CLR/HOME e rivisualizzate il programma battendo LIST.

13. Ora battete

```
125 LET D=B-A <RETURN>
135 PRINT D <RETURN>
```

Cancellate lo schermo e visualizzate il programma. Che cosa è successo?

....

14. Studiate con calma il programma. Che cosa accadrà se eseguirete con RUN il programma?

....

Battete RUN, premete il tasto RETURN, e registrate qui sotto quello che il computer ha fatto.

....

15. Nel programma originale i numeri di riga non erano consecutivi (ad es.: 100, 101, 102, 103, ecc.) ma avevano degli intervalli (ad es. 100, 110, 120, 130 e 140). Potete pensare ora al motivo per cui è stato fatto ciò? (Suggerimento: vedere il punto 13).

....

16. Come si fa ad inserire delle righe in un programma BASIC? (Suggerimento: vedere i punti 13 e 15).

....

17. Cancellate il programma dalla memoria battendo NEW e premendo il tasto RETURN. Immettete il programma seguente

```
100 INPUT BIANCO <RETURN>
110 LET ROSSO=BIANCO+2 <RETURN>
120 PRINT ROSSO <RETURN>
130 GOTO 100 <RETURN>
140 END <RETURN>
```

18. Questo nuovo programma ha varie caratteristiche che non avete visto prima. Studiate attentamente il programma e pensate a quello che accadrebbe se lo eseguiste con RUN. INPUT in inglese vuol dire "immetti"; sapendo che GOTO significa in inglese "vai a", che cosa significa il GOTO 100 nella riga 130?

   ....

19. Eseguite con RUN il programma e registrate quello che il computer ha fatto.

   ....

   Battete il numero 6 e premete il tasto RETURN. Che cosa è successo?

   ....

20. Battete il numero 10 e premete il tasto RETURN. Che cosa è successo?

   ....

21. Quale riga del programma pensate che generi il punto interrogativo?

   ....

   Descrivete con parole vostre quello che il programma fa. Se necessario, fate altre prove per essere sicuri di non sbagliare.

   ....

22. Ora vogliamo uscire dal programma. Individuate i tasti RUN/STOP e RESTORE (situati all'estremità sinistra e destra della tastiera) e premeteli contemporaneamente. D'ora in poi citeremo questi tasti come RUN/STOP/RESTORE. Che cosa è successo?

   ....

   Se non è successo niente, riprovate.

23. Battete NEW per cancellare il programma in memoria. Battete il seguente programma:

```
100 LET A=100 <RETURN>
110 PRINT A <RETURN>
120 LET A=A+1 <RETURN>
130 GOTO 110 <RETURN>
140 END <RETURN>
```

Eseguite il programma e scrivete cosa è successo.

....

Quando vi stancate di guardare il video, premete RUN/STOP. Che cosa è accaduto?

....

24. Provate ancora una volta. Eseguite con RUN il programma e dopo che qualche numero è stato visualizzato, interrompetelo. Come si fa a interrompere un programma BASIC che è in esecuzione sul computer?

....

25. Cancellate lo schermo e visualizzate il programma in memoria. (Vedere i punti 4 e 5). Battete le righe sottostanti. Si noti l'assenza di spazi nella prima riga e gli spazi in più nella seconda.

```
100LETA=1 <RETURN>
1 2 0 L E T A = A + 1 <RETURN>
```

Cancellate lo schermo e listate il programma. Che cosa è successo?

....

Eseguite il programma. Cosa è successo?

....

Ora battete RUN/STOP per interrompere il programma. È chiaro che alcuni spazi sono importanti nelle istruzioni BASIC e altri no. Per ora vi basterà notare il fatto. Torneremo su questo argomento più tardi.

26. Proviamo un programma con qualche nuova caratteristica. Cancellate il programma dalla memoria battendo NEW e premendo quindi il tasto RETURN. Immettete il programma qui sotto riportato.

```
100 PRINT "BATTERE UN NUMERO" <RETURN>
110 INPUT PRIMO <RETURN>
120 PRINT "ANCORA UNA VOLTA"
130 INPUT SEC <RETURN>
140 LET SOMMA=PRIMO+SEC <RETURN>
150 PRINT "LA LORO SOMMA E´" <RETURN>
160 PRINT SOMMA <RETURN>
170 END <RETURN>
```

27. Studiate per qualche istante il programma. Ora eseguitelo con RUN. Che cosa è successo?

    ....

    Battere il numero 12, premere il tasto RETURN, e registrare qui sotto quello che il computer ha fatto.

    ....

28. Bene, ora battete il numero 13, premete il tasto RETURN, e registrate qui sotto quello che è successo.

    ....

29. Vogliamo cambiare "ANCORA UNA VOLTA" in "ANCORA UN NUMERO" nella riga 120. Certamente potremmo ribattere l'intera riga, ma possiamo sfruttare l'editing del BASIC C-64.
    Battete

```
LIST 120 <RETURN>
```

Cosa è successo?

....

Premete SHIFT e il tasto contrassegnato da ⇑CRSR⇓ sul lato destro in basso. Cosa è successo?

....

Premete SHIFT/⇑CRSR⇓ ancora due volte. Il cursore dovrebbe ora trovar-

si sulla riga 120. Senza lo SHIFT premuto, premete ⇐CRSR⇒ diverse volte fino a che il cursore è sulle seconde virgolette. Premete DELETE 7 volte per cancellare le lettere A T L O V (spazio) A alla sinistra del cursore. Ora battete (spazio) NUMERO" e premete RETURN. Cosa è successo?

....

Listate il programma. La linea 120 dovrebbe essere

```
120 PRINT "ANCORA UN NUMERO"
```

30. Se vi accorgete che la riga non è stata cambiata correttamente, ribattetela o ricorreggetela.

31. Si può cambiare una riga, dopo averla listata, muovendo il cursore CRSR fino alla posizione dove si devono effettuare cambiamenti e ribattendo le correzioni. RETURN ricolloca la riga cambiata nel programma.

32. Facciamo un altro cambiamento nella riga 120. Listatela, muovete il cursore fino alle seconde virgolette alla fine della riga. Premete SHIFT e INST/DEL una volta. Cosa è sucesso?

....

33. Ora premete SHIFT/INST/DEL 6 volte. Inserite 7 caratteri

```
, PREGO
```

34. Ora premete RETURN per completare l'editing della riga. Visualizzate il programma per vedere se sono stati inseriti i cambiamenti. La riga 120 dovrebbe essere:

```
120 PRINT "ANCORA UN NUMERO, PREGO"
```

35. Eseguite il programma. Alle domande di input, battete 22 e 23. Il programma ha un aspetto migliore?

....

36. Consideriamo ora un argomento diverso. Cancellate lo schermo. Battete NEW e premete il tasto RETURN per eliminare il programma dalla memoria. Immettete quindi il programma seguente.

```
100 LET A=1 <RETURN>
110 LET A$="CASA" <RETURN>
120 PRINT A <RETURN>
130 PRINT "A" <RETURN>
140 PRINT A$ <RETURN>
150 PRINT "A$" <RETURN>
160 END <RETURN>
```

37. Questo programma contiene qualcosa di nuovo. Osservate l'A$ nella riga 110. Si noti che è posto uguale ad una parola racchiusa fra virgolette. Il programma ha a che fare con variazioni sulla stampa di A e A$. Eseguite con RUN il programma e registratene l'uscita.

....

38. Si studi attentamente l'uscita e si identifichi quello che è stato stampato in risposta ad ognuna delle istruzioni PRINT. Per ora fate solo i confronti. Più tardi esamineremo l'argomento più dettagliatamente. Immettere la riga seguente:

```
155 PRINT B <RETURN>
```

39. Cancellate lo schermo e visualizzate il programma con il comando LIST. Notate che l'unico posto in cui B viene menzionato è la riga 155 nell'istruzione PRINT. Cosa pensate che accadrà se facciamo eseguire il programma?

....

Bene, ora eseguite con RUN il programma e registrate quello che avviene.

....

40. Come si è visto nel Capitolo 1, anche se il valore di B non è stato definito nel programma, il computer gli ha assegnato il valore zero. Questo è un fatto importante da ricordare mentre si scrivono programmi.

41. Con questo si concludono le attività di scoperta per questo capitolo. Spegnete il computer e la TV e andate avanti con la sezione successiva.

2.3 Analisi

CORREGGERE I PROGRAMMI

Siccome la maggior parte di noi fa errori quando batte a macchina, dobbiamo essere in grado di correggerli. Supponiamo che sia stato fatto un errore mentre si batteva una riga. Se non avete già premuto il tasto RETURN alla fine della riga potete spostare il cursore a sinistra usando il tasto DELETE e apportare le necessarie correzioni come si è visto nel Capitolo 1. Quando in una riga sono state effettuate tutte le correzioni, premete il tasto RETURN.
Se si vuole cambiare una riga dopo che è stato premuto il tasto RETURN si può usare l'editing del C-64. Per esempio, se volete cambiare la riga 110 di un programma, dovete battere

```
LIST 110 <RETURN>
```

Il computer visualizzerà la riga 110; usate i tasti del movimento del cursore in basso a destra ⇐CRSR⇒ ⇓CRSR⇑ per posizionarlo dove devono essere effettuati i cambiamenti. Battete sopra i caratteri esistenti o usate INST/DEL o SHIFT/INST/DEL per correggere. Premete RETURN quando avete finito di correggere la riga. Gli spazi inseriti con SHIFT/INST/DEL devono essere cancellati (col tasto DELETE) prima degli altri caratteri. Dopo che è stato premuto SHIFT/INST/DEL una volta e si preme INST/DEL, appare una T inversa.
Premendo ancora INST/DEL si cancellerà questo simbolo.

REQUISITI DEI PROGRAMMI IN BASIC

Abbiamo visto vari aspetti importanti dei programmi in BASIC. Come esempio torneremo al programma usato precedentemente nel lavoro sul computer:

```
100 LET A=1
110 LET B=8
120 LET C=A+B
130 PRINT C
140 END
```

Ogni programma in BASIC è formato da un gruppo di righe chiamate istruzioni. Ciascuna istruzione deve avere un numero di riga e non deve superare gli 80 caratteri. Nel suddetto programma ci sono tre tipi di istruzioni BASIC: LET, PRINT e END. Le prime due istruzioni saranno

trattate completamente nei capitoli seguenti. Per ora l'uso di ciascuna di queste istruzioni nel programma è chiaro. L'istruzione END ("fine") non è richiesta dal Commodore 64. Il programma si potrà eseguire con o senza l'istruzione END. È però buona norma usarla, specialmente in programmi lunghi, in modo che non ci sia confusione su dove il programma termina. Ci faremo una regola di usare l'istruzione END in tutti i programmi, anche se il computer non la richiede.
Generalmente i numeri di riga in un programma BASIC non sono numerati consecutivamente (come 100, 101, 102, ecc.). Il motivo è che potremmo aver bisogno in seguito di inserire delle istruzioni addizionali se scopriamo errori o vogliamo modificare il programma. Se le righe fossero numerate consecutivamente e fossero distanziate di 1 fra di loro, sarebbe meno facile fare i cambiamenti. Con degli intervalli fra i numeri di riga, possono essere inserite nuove istruzioni con numeri di riga che non si trovino già nel programma.
Come abbiamo visto negli esercizi di scoperta, alcuni spazi sono importanti nei programmi BASIC. Sarebbe tempo sprecato soffermarci su questo argomento. È bene però che teniate il problema degli spazi nel fondo della vostra mente come una possibile soluzione degli errori che possono capitare. Basti dire che, se rendete le istruzioni dei vostri programmi leggibili, non dovreste avere dei problemi con gli spazi.
Potete immettere le righe del programma in ordine casuale. Se battiamo per esempio

```
140 END
120 LET C=A+B
110 LET B=8
130 PRINT C
100 LET A=1
```

e questo nuovo programma viene listato, il computer metterà in ordine le istruzioni e le visualizzerà in ordine di numero. Allo stesso modo, se dovessimo eseguire il programma battuto in quel modo, il computer metterebbe in ordine le istruzioni nel modo appropriato prima di iniziare l'esecuzione.
Si può togliere un'istruzione BASIC dal programma battendo il numero di riga e premendo il tasto RETURN. Le istruzioni possono essere modificate ribattendo le righe interessate e premendo RETURN dopo che ciascuna riga è stata battuta, oppure usando l'editing. Abbiamo visto infine che si possono aggiungere istruzioni usando numeri di riga non ancora utilizzati.

DIRE AL COMPUTER QUEL CHE DEVE FARE

Occorre fare una distinzione netta fra le istruzioni di un programma BASIC e i comandi. I comandi dicono al computer di fare qualcosa con un programma. Nel lavoro sul computer ne abbiamo visti diversi ed ora ne ripasseremo brevemente l'uso.
Quando un programma BASIC viene immesso, va nella memoria. Spesso c'è bisogno di vedere il programma contenuto in memoria per controllare i cambiamenti fatti o farne una copia. In ogni caso, già sapete che questo si fa con il comando LIST. Questo comando è molto utile. Se un programma non funziona come dovrebbe, il primo passo da fare è quello di visualizzarlo. Il modo di risolvere il problema è quello di far sì che il computer fornisca una copia del programma che sta eseguendo. Usate questa copia per cercare l'errore nel programma.
Quando si spegne il computer, il contenuto della memoria si cancella automaticamente. Mentre si lavora sul computer, però, è possibile avere programmi che si mescolano l'un con l'altro senza che ce ne accorgiamo. Supponete di star lavorando con un programma e di decidere di proseguire con un altro. Se prima non cancellate il primo programma dalla memoria, il secondo entrerà sopra il primo col risultato che parti di entrambi i programmi si mescoleranno in memoria. Per evitare questo state attenti a cancellare un programma con il comando NEW quando avete finito di lavorare con esso.
Un programma BASIC è semplicemente un gruppo di istruzioni che devono agire sul computer. Tuttavia il computer ha bisogno che gli si dica di cominciare il procedimento. Quando riceve il comando RUN, il computer va alla riga che nel programma ha il numero più basso, esegue le istruzioni, passa alla riga con il numero immediatamente superiore, e continua ad eseguire istruzioni in ordine numerico, a meno che il programma indichi che un'istruzione debba essere eseguita al di fuori dell'ordine consecutivo.

Usare SHIFT/CLR/HOME per pulire lo schermo. Poi battere LIST per visualizzare il programma in memoria

Battere NEW per cancellare il programma in memoria

Battere RUN per far eseguire il programma

IMMISSIONE E CONTROLLO DEI PROGRAMMI

Fino ad ora, quando vi è stato detto di immettere dei programmi o dei comandi, vi è stata data la scritta <RETURN> per ricordarvi di premere il

relativo tasto. Quest'abitudine dovrebbe essere ben sviluppata ora, per cui in futuro non useremo più la scritta <RETURN>. D'ora in poi quando avete finito di battere un'istruzione o un comando, premete il tasto RETURN per far sapere al computer che avete finito.
Possono sorgere delle situazioni in cui si ha bisogno di controllare un programma in esecuzione. Certamente uno dei casi più drammatici è quando un programma si trova in un'iterazione chiusa (loop) e continuerebbe a girare per sempre se non lo interrompessimo. Possiamo interrompere un programma di questo tipo tenendo abbassato RUN/STOP: il computer interrompe l'esecuzione del programma e ci dice BREAK IN ("interrotto a") seguito dal numero della riga che stava eseguendo al momento dell'interruzione.
Una situazione diversa capita quando il computer è in un'iterazione di ingresso (input loop) in attesa che sia immesso un dato. Se vogliamo uscire da una situazione di questo tipo, premiamo RUN/STOP/RESTORE. Il computer allora salta fuori dall'esecuzione del programma tornando di nuovo nel modo READY.

NOMI DELLE VARIABILI IN BASIC

Uno degli aspetti del BASIC che più spesso causano problemi ai principianti, riguarda i nomi delle variabili e la distinzione fra il nome della variabile e i dati memorizzati sotto quel nome. Nell'istruzione BASIC

```
100 LET A=2
```

A è il nome di una variabile. Con "variabile" vogliamo dire che diversi valori possono essere assegnati ad A. Di conseguenza, le istruzioni LET sono spesso chiamate istruzioni di assegnamento. In questo caso alla variabile A viene assegnato il valore 2. In realtà quello che ha luogo è che, da qualche parte nel computer, c'è una posizione di memoria chiamata A, e che il numero 2 è memorizzato in quella posizione. L'idea fondamentale è quella di separare il nome di una posizione in memoria dal contenuto della posizione stessa. È la stessa differenza che passa tra il numero di una casella in un ufficio postale e il contenuto di quella casella. Il numero della casella non cambia, ma il contenuto può essere cambiato ogni momento.
Considerate la seguente riga di istruzione:

```
130 LET C=A+B
```

Questa istruisce il computer a prendere i numeri immagazzinati nelle posizioni chiamate A e B, sommarli assieme e mettere la somma nella posi-

zione di immagazzinamento chiamata C. Il segno di uguale significa che si deve calcolare quello che si trova a destra e assegnarlo al nome di variabile sulla sinistra. Supponiamo ora di avere un'istruzione BASIC come

```
120 LET B=B+1
```

Se consideriamo la riga suddetta come un'equazione algebrica, abbiamo

$$B = B+1$$

Se si sottrae B da entrambi i lati di questa equazione, abbiamo

$$0 = 1$$

il che è davvero molto strano! È chiaro che in un'istruzione BASIC il segno = non significa la stessa cosa che in un'equazione algebrica. Invece, l'istruzione

```
120 LET B=B+1
```

istruisce il computer a prendere il numero situato nella posizione B, ad aggiungere 1 a quel numero e a mettere il risultato di nuovo nella posizione di immagazzinamento chiamata B.
Se mettiamo un numero in una posizione di memoria, tutto ciò che vi era memorizzato prima viene perso. Si considerino le seguenti istruzioni:

```
100 LET A=1
110 LET A=2*3
```

La riga 100 istruisce il computer a stabilire una posizione chiamata A e a mettere il numero 1 in quella posizione. La riga 110 dice al computer di moltiplicare 2 per 3 e memorizzare il prodotto nella posizione A. L'1 memorizzato in precedenza nella posizione A è stato perso.
Questo ci porta al cuore del problema. La lettera A, che identifica la posizione, si chiama variabile perché il contenuto di A può essere cambiato. Il nome della posizione non cambia, ma può esserlo il numero che vi è memorizzato.
Per essere più precisi, la variabile A cui si fa riferimento sopra si chiama variabile "numerica". Il motivo per cui si è aggiunto l'aggettivo "numerica" al nome è che c'è un altro tipo di variabile, chiamata "stringa di caratteri". Nelle attività di scoperta è stato brevemente introdotto questo concetto; per quanto riguarda i nomi, è facile distinguere le variabili numeriche da quelle a stringa di caratteri. A, LIBRO, M e P identificherebbero tutte delle variabili numeriche e darebbero il nome a quantità nu-

meriche. A$, LIBRO$, M$ e P$ indicano tutte stringhe di caratteri. Il simbolo $ che vi è aggiunto identifica il nome come una variabile a stringa di caratteri. Nell'istruzione BASIC

```
100 LET B$="GRANAIO"
```

B$ dà il nome ad una posizione in memoria nella quale è memorizzata la stringa di caratteri "GRANAIO". Le virgolette delimitano la stringa, ma non ne fanno parte.
Ricordate che il BASIC C-64 ha delle regole molto elastiche per i nomi delle variabili. Vi permette di usare dei nomi lunghi sia per le variabili numeriche che per quelle a stringa di caratteri: si possono usare fino a 77 caratteri (compreso il carattere $ nel caso di stringhe di caratteri). Il computer tuttavia distingue le variabili solo dai primi due caratteri del loro nome; esso ha inoltre un gruppo di parole "riservate" che sono usate in BASIC e per comandi. Queste parole non possono essere usate per dare il nome a delle variabili. Però, se fate un errore e ne usate una, il computer ve lo farà sapere!
Per riassumere, un nome di variabile in BASIC identifica una posizione di immagazzinamento in memoria che può contenere un numero o una stringa di caratteri. Il contenuto della posizione di memoria (il valore della variabile) può essere cambiato, ma il nome della posizione non può esserlo.
L'istruzione LET (assegnamento) calcola quello che si trova alla destra del segno uguale e lo assegna alla posizione di memoria il cui nome si trova alla sinistra. Perciò,

```
100 LET D=A+B+C
```

istruisce il computer a calcolare l'espressione (A+B+C) usando i numeri memorizzati nelle posizioni di memoria chiamate A, B, e C. Il risultato è quindi memorizzato nella posizione di memoria chiamata D.
L'uso di LET nelle istruzioni di assegnamento è facoltativo nel BASIC C-64; per motivi di coerenza, in questo libro lo useremo sempre.
Abbiamo appena grattato la superficie per quanto riguarda le variabili a stringa di caratteri. Torneremo su quest'argomento varie volte nel corso del manuale.

2.4 Test di apprendimento

1. Come segnalate al computer che avete finito di battere una riga o un'istruzione?

   ....

2. Supponete che il vostro computer stia aspettando che voi immettiate un numero per un'istruzione di ingresso (INPUT) in un programma. Invece voi decidete di uscire dal programma. Come fate?

   ....

3. Come si fa a fermare un programma in esecuzione sul computer?

   ....

4. Che cosa c'è di sbagliato nel programma seguente?

```
100 LET A=1
110 LET B=3
120 LET C=B-A
PRINT C
130 END
```

   ....

5. Che cosa succederebbe se il programma del punto 4 venisse corretto e fosse avviato con RUN?

   ....

6. Quanto possono essere lunghi i nomi delle variabili?

   ....

7. Come si fa ad inserire una riga in un programma BASIC?

   ....

8. Come si sostituisce una riga in un programma BASIC?

   ....

9. Come si fa a togliere una riga da un programma BASIC?

   ....

10. Come si visualizza il programma in memoria?

    ....

11. Come si fa a cancellare lo schermo?

   ....

12. Come si fa a comandare al computer di eseguire il programma in memoria?

   ....

13. Come si cancella un programma in memoria?

   ....

14. Qual è la differenza fra una variabile numerica e una variabile a stringa di caratteri?

   ....

15. Che cosa fa il seguente comando?

```
LIST 120
```

   ....

16. Quali tasti muovono il cursore?

   ....

17. Quali tasti si usano per inserire e cancellare caratteri in una riga?

   ....

18. Quando una riga è stata corretta, come bisogna terminare la correzione?

   ....

Capitolo 3

Aritmetica col computer Gestione dei programmi

3.1 Obiettivi

Per alcuni esercizi di questo capitolo avrete bisogno di un disco vuoto. Le istruzioni per utilizzare il drive sono date nel *VIC-1541 Single Drive Floppy Disk User's Manual*.

ARITMETICA COL COMPUTER

Se si va a vedere fino in fondo, tutta la matematica che si fa col computer usa solo le operazioni aritmetiche più semplici. È essenziale avere una visione chiara di come queste operazioni vengono fatte.

PARENTESI NEI CALCOLI

Tutte le espressioni matematiche devono essere battute in una sola istruzione per essere immesse nel computer. Alcune espressioni possono essere trattate in questo modo solo se organizzate in parentesi. L'uso efficace delle parentesi diventa così una necessità.

NOTAZIONE ESPONENZIALE

Il computer ha a che fare sia con numeri molto grandi sia con numeri molto piccoli. Per descriverli utilizza la notazione E (esponenziale). Dobbiamo quindi essere in grado di riconoscerla e interpretarla.

FORMATTARE UN DISCHETTO

I dischi nuovi o vuoti devono essere preparati o "formattati" per essere usati sul Commodore 64. Dobbiamo sapere come si formatta correttamente un disco per potervi memorizzare i programmi.

ARCHIVIAZIONE E RICHIAMO DI PROGRAMMI

Saranno introdotti ora i comandi che ci permetteranno di archiviare e richiamare i programmi usando un disco. Saranno inoltre esplorati i comandi per la gestione dei programmi.

3.2 Esercizi di scoperta

Per installare il drive seguite le seguenti istruzioni: per prima cosa spegnete il computer; inserite il cavo grigio fornito con il drive nell'apposita presa sul retro del drive e nella presa di corrente. Assicuratevi che il drive sia spento (OFF) e che quindi le spie luminose siano spente.
Effettuate il collegamento tra il drive e il computer inserendo l'apposito cavo nella porta seriale a sinistra sul retro del drive e in quella a destra sul retro del C-64. Accendete prima il drive e poi il computer (il computer sempre dopo!). Troverete maggiori dettagli nel manuale allegato al drive. Faremo uso del drive dall'esercizio n. 11 in poi.

1. Accendete il computer e la TV. Ricordate dal Capitolo 2 i seguenti simboli per le operazioni aritmetiche:

   \+ Addizione
   − Sottrazione
   * Moltiplicazione
   / Divisione

   Per ripassare le operazioni aritmetiche, battete il seguente programma:

```
100 INPUT A
110 INPUT B
120 LET C=A*B-B/3
130 PRINT C
140 END
```

Visualizzate il programma e studiatelo brevemente. Se si esegue il programma e si immette 2 per A e 3 per B, che cosa pensate che verrà stampato?

....

Eseguite il programma immettendo 2 e 3 e scrivete quello che è successo.

....

2. Cancellate il programma in memoria. Ora battete

```
100 LET A=3*3
110 LET B=3↑2
120 PRINT A
130 PRINT B
140 END
```

Visualizzate il programma e assicuratevi che sia corretto. Ora eseguitelo con RUN e registrate qui sotto che cosa viene stampato.

....

Confrontate i numeri stampati con le espressioni che si trovano nelle righe in cui questi sono stati calcolati. Cercate di capire quello che è avvenuto.

3. Battete

```
100 LET A=3*3*3
110 LET B=3↑3
```

Avviate il programma. Che cosa viene stampato?

....

4. Battete

```
100 LET A=2*2*2*2
110 LET B=2↑4
```

Avviate con RUN il programma. Che cosa viene stampato?

....

A che cosa serve il simbolo ↑?

....

5. Ricordate dalle vostre reminiscenze di algebra (se non conoscete l'algebra, non disperatevi!) che, quando un numero dev'essere moltiplicato per se stesso, per esempio tre volte, possiamo indicare ciò con un esponente. Se il numero fosse 2, potremmo scrivere l'espressione come

$$2^3$$

Come sarebbe scritta quest'espressione in BASIC usando il simbolo ↑? (Suggerimento: vedere i punti 2, 3 e 4).

....

6. Scrivete gli operatori (simboli) che richiamano le seguenti operazioni aritmetiche:

Divisione

....

Elevazione a potenza

....

Moltiplicazione

....

7. Cancellate il programma in memoria. Battete

```
100 LET A=4+2*6
110 LET B=(4+2)*6
120 LET C=4+(2*6)
130 PRINT A
140 PRINT B
150 PRINT C
160 END
```

I due punti importanti di questo programma sono l'ordine in cui vengono eseguite le operazioni aritmetiche e l'effetto delle parentesi. Se guardate con attenzione, è chiaro che in ognuno dei calcoli delle righe 100, 110 e 120 sono coinvolti gli stessi numeri e le stesse operazioni. L'unica differenza è data dal raggruppamento con parentesi nelle righe. Avviate il programma e registrate quello che viene visualizzato.

....

Ora cambiate le righe 100, 110 e 120:

```
100 LET A=4+2*6/3
110 LET B=(4+2)*6/3
120 LET C=4+(2*6)/3
```

Eseguite il programma e scrivete i numeri visualizzati.

....

Studiate il programma e i numeri che il computer ha presentato fino a quando capite quello che avviene nel programma. Ci sono delle regole molto specifiche che il computer usa in tali situazioni. Esamineremo queste regole più avanti in questo capitolo.

....

8. Cancellate il programma in memoria. Ora immettete il seguente programma:

```
100 LET A=3*100*100
110 LET B=3*100*100*100
120 LET C=3*100*100*100*100
130 LET D=3*100*100*100*100*100
140 PRINT A
150 PRINT B
160 PRINT C
170 PRINT D
180 END
```

Eseguite il programma e scrivetene qui sotto l'uscita.

....

Sapete spiegare le varie forme in cui i numeri sono stati presentati? (Suggerimento: contate il numero degli zeri nei moltiplicatori delle righe 100, 110 e 120).

9. Battete

```
100 LET A=4/(100*100)
110 LET B=4/(100*100*100)
120 LET C=4/(100*100*100*100)
125 LET E=4/(100*100*100*100*100)
165 PRINT E
```

Eseguite il programma e registratene qui sotto l'uscita.

....

Ancora, avete capito quello che è successo? Contate gli zeri nei denominatori delle righe 100, 110, 120 e 125.

10. Se in un numero presentato dal computer appare una E, che cosa significa? Spiegatelo con parole vostre.

....

Se non capite ancora completamente lo scopo della notazione **E**, non preoccupatevi. Ci torneremo sopra più avanti.

11. Passiamo ora ad esaminare come si archiviano e richiamano i programmi dal dischetto. Prima però dovete imparare a formattare un disco. (Se non avete ancora installato il drive, seguite le istruzioni del manuale. Non dimenticate di spegnere il computer prima di installare il drive). Se volete invece registrare il programma su cassetta, leggete le istruzioni nella *Guida di riferimento per il programmatore*.

12. Spegnete il computer e accendete il drive; ora riaccendete il computer. Inserite il dischetto nel drive con la tacca sulla sinistra e l'etichetta verso l'alto. Chiudete lo sportellino del drive. Con il dischetto inserito battete (ma non premete RETURN):

```
OPEN 15,8,15,"NEW0:BASIC,ZZ"
```

Nota: Tutti i programmi e le informazioni su disco vanno persi se il dischetto è formattato premendo RETURN.

Ora premete RETURN.

13. Si accenderà la spia rossa sul drive e sullo schermo apparirà READY. Il drive "ronzerà" per circa 70 secondi. Quando il rumore cesserà e la spia sarà spenta sarete pronti ad utilizzare il disco. Non procedete fino a quando la luce rossa non si spegne.

14. Cancellate la memoria e battete il seguente programma

```
100 LET A=2
110 LET B=3
120 LET C=A*B
130 PRINT C
140 END
```

15. Visualizzate il programma per assicurarvi che sia giusto, ed eseguitelo.

16. Ora battete

```
SAVE"PRODOTTO",8 <RETURN>
```

Cosa è successo?

....

Così facendo si archivia il programma in memoria sul disco sotto il nome PRODOTTO. (Potete scegliere per un programma qualsiasi nome lungo al massimo 16 caratteri.)

17. Ora battete

```
LOAD"$",8 <RETURN>
```

per caricare il catalogo ($) del dischetto nel drive (8). Quando appare READY battete

```
LIST <RETURN>
```

per visualizzare il catalogo. Leggerete

```
0 "BASIC                " ZZ 2A
1    "PRODOTTO"         PRG
663 BLOCKS FREE.

READY.
```

Il programma **PRODOTTO** è ora sul dischetto. PRG sta per "programma". Il numero alla sinistra (1) indica la misura in blocchi (256 caratteri) del programma **PRODOTTO**. La linea *nera* indica che il dischetto è stato intitolato **BASIC**, il suo **ID** è **ZZ** e la versione del sistema operativo è **2A**. (Confrontate con l'istruzione **OPEN** al punto 12).

18. Cancellate la memoria battendo

```
NEW <RETURN>
```

Richiamate di nuovo l'elenco dei programmi sul disco. (Vedere il punto 17). I programmi sul disco sono stati cambiati in qualche modo quando il programma in memoria è stato cancellato?

....

19. Battete **NEW** per cancellare il catalogo dalla memoria e battete il programma seguente

```
100 LET D=2*6-8/4
110 PRINT D
120 END
```

Eseguite il programma. Pensate ad un nome per questo programma che sia diverso da **PRODOTTO**; limitate il nome a 16 lettere dell'alfabeto o meno. Scrivete il nome qui sotto.

....

Spostate il programma in memoria sul disco sotto il nome che avete appena scritto qui sopra. (Vedi il punto 16). Ora visualizzate il programma in memoria. Il programma che avete archiviato or ora su disco è ancora in memoria?

....

20. Richiamate l'elenco dei programmi archiviati sul disco. (Vedi il punto 17). Vi sono i due programmi che avete appena immesso?

....

Dovremmo avere archiviato due programmi che sono stati appena immessi. Il nome del primo programma è PRODOTTO; il nome del secondo è stato scritto al punto 19. Per semplificare la discussione, chiameremo questo secondo programma PROGRAMMA 19. Voi, naturalmente, dovrete usare il nome che avete scelto per il secondo programma.

21. Per spostare PRODOTTO dal disco alla memoria battete

```
LOAD"PRODOTTO",8 <RETURN>
```

Visualizzate questo programma e verificate che sia quello giusto. Aggiungete la riga 125 al programma in memoria:

```
125 PRINT "IL PRODOTTO E'"
```

Visualizzate il programma e fatelo girare. Per sostituire il vecchio programma PRODOTTO sul dischetto con quello modificato in memoria, battete

```
SAVE"@0:PRODOTTO",8 <RETURN>
```

22. Cancellate il programma in memoria. Battete LIST per assicurarvi che non ci sia alcun programma in memoria. Ora spostate PRODOTTO dal dischetto alla memoria e visualizzatelo. Spostate il PROGRAMMA 19 dal dischetto alla memoria. Visualizzate il programma in memoria. Quale dei due appare?

....

Che cosa è successo al programma PRODOTTO che si trovava in memoria quando abbiamo spostato il PROGRAMMA 19 in memoria dal disco?

....

23. Spostate PRODOTTO in memoria col comando LOAD. (Vedere il punto 21). Togliete il programma PRODOTTO dal disco battendo

```
CLOSE 15
OPEN 15,8,15,"SCRATCH0:PRODOTTO"
```

Visualizzate il programma in memoria. Quando abbiamo cancellato

PRODOTTO dal disco, questo fatto ha modificato qualcosa in memoria?

....

24. Richiamate l'elenco dei programmi archiviati sul disco. PRODOTTO non dovrebbe essere compreso nell'elenco, ma PROGRAMMA 19 vi si dovrebbe trovare. Esaminate l'elenco dei programmi. È tutto come dovrebbe essere? (Vedi il punto 17).

....

25. Si noti che se ora cancelliamo la memoria, perderemo sia la copia di PRODOTTO che era sul disco (l'abbiamo tolta al punto 23), che la copia in memoria. Cancellate la memoria. (Vedi il punto 18). Visualizzate il programma in memoria. È rimasto qualcosa in memoria?

....

Provate a spostare PRODOTTO dal disco alla memoria. (Vedi il punto 21). Che cosa è successo?

....

Se la luce rossa del drive lampeggia è stato commesso un errore. I comandi SAVE e LOAD interromperanno la segnalazione d'errore.
Cancellate PROGRAMMA 19 dal disco. (Vedi il punto 23). Togliete il disco dal drive, spegnete il computer, la TV e il drive.

3.3 Analisi

ARITMETICA COL COMPUTER

Esaminiamo le cinque operazioni aritmetiche. Queste sono l'addizione, la sottrazione, la moltiplicazione, la divisione e l'elevazione a potenza (+, −, *, /, ↑).
L'operazione di elevazione a potenza è rappresentata dal simbolo ↑. Perciò 3↑5 significa "3 elevato alla quinta potenza", che a sua volta significa 3 moltiplicato per se stesso quattro volte, con il risultato di 243.
Dobbiamo stare molto attenti a capire l'ordine in cui le operazioni aritmetiche sono eseguite dal computer. Un esempio servirà ad illustrare

questo punto. Consideriamo la seguente espressione

$$2+3\uparrow 2/5-1$$

Se il computer esaminasse semplicemente l'espressione da sinistra a destra, eseguendo le operazioni man mano che le incontra, il risultato sarebbe 2 più 3 (che fa 5), elevato alla seconda potenza (che fa 25), diviso per 5 (che fa 5) meno 1, dando una risposta di 4. Supponiamo però che l'addizione e la sottrazione siano eseguite per prime, poi l'elevazione a potenza, e poi la moltiplicazione e la divisione. Questo darebbe 5 elevato alla seconda potenza diviso per quattro. E quindi 5 elevato alla seconda potenza (che fa 25) diviso per quattro dà come risposta 6,25.
Potremmo continuare applicando regole diverse per l'ordine delle operazioni aritmetiche e ottenere ogni volta risposte diverse. Il fatto importante è che in BASIC ci sono regole ben definite per l'ordine e la priorità delle operazioni aritmetiche, e noi dobbiamo capirle.
L'ordine delle operazioni è da sinistra a destra usando le regole di priorità qui sotto esposte.

La priorità per le operazioni aritmetiche è
1. elevazione a potenza
2. moltiplicazione e divisione
3. addizione e sottrazione

Ora torniamo al nostro esempio di

$$2+3\uparrow 2/5-1$$

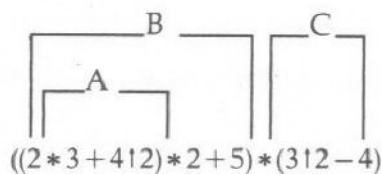
ed esploriamo da sinistra a destra alla ricerca di un'elevazione a potenza. Dal momento che è indicata un'elevazione a potenza (3↑2), questa verrà eseguita per prima. Ora l'espressione è

$$2+9/5-1$$

Esplorando ancora da sinistra a destra, cerchiamo una elevazione a potenza e, non trovandone nessuna, cerchiamo le operazioni che hanno la priorità successiva (moltiplicazione e divisione). Subito dopo viene quindi eseguita la divisione, col seguente risultato:

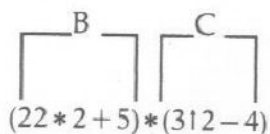
$$2+1.8-1$$

Siccome non sono rimaste altre moltiplicazioni o divisioni nell'espressione, esploriamo da sinistra a destra alla ricerca di addizioni e sottrazioni.

L'addizione dà

$$3.8 - 1$$

e la sottrazione finale ci dà la risposta: 2.8.
Ripassate le regole per l'ordine e la priorità delle operazioni aritmetiche fino a che queste diventino parte di voi stessi. Osserveremo di nuovo queste regole nella prossima sezione.

PARENTESI NEI CALCOLI

Le regole per l'ordine e la priorità delle operazioni aritmetiche non sono però tutto sull'argomento. Per rendercene conto, consideriamo il seguente esempio un po' più complicato:

```
 ┌────────B────────┐ ┌──C──┐
 │ ┌───A────┐      │ │     │
((2*3+4↑2)*2+5)*(3↑2-4)
```

La differenza fra questa espressione e quelle che abbiamo studiato è l'uso delle parentesi per raggruppare delle parti dell'espressione. Analizzeremo questo esempio molto a fondo per mostrarvi come il computer affronti l'aritmetica.
Il computer inizia esplorando da sinistra a destra e incontra la parentesi sinistra di B. Guarda dentro per vedere se ci sono altre parentesi sinistre e ne trova una per A. La parentesi successiva che incontra è la parentesi destra di A. A questo punto il computer ha isolato il primo gruppo di operazioni da fare nella parentesi A:

```
2*3+4↑2
```

e lo calcola usando le regole sull'ordine e la priorità. Il risultato è 22 (controllate). Ora il nostro problema è diventato

```
┌───B───┐ ┌──C──┐
(22*2+5)*(3↑2-4)
```

Nella esplorazione successiva il computer isola le parentesi B, esegue i calcoli aritmetici che si trovano all'interno, e il problema è ora

C

49*(3↑2−4)

Siccome restano solo le parentesi C, il computer esegue le operazioni che vi si trovano all'interno, dando

49*5

che, dopo la moltiplicazione finale, dà come risultato 245.
In conclusione, se le parentesi sono annidate, il computer esegue i calcoli partendo dal paio di parentesi più interne, lavorando da sinistra a destra. Quando un gruppo di parentesi è stato eliminato, le operazioni aritmetiche al suo interno vengono eseguite secondo le regole sull'ordine e la priorità già date. Una buona regola da seguire è che, se vi può essere qualche confusione circa il modo in cui il computer calcola un'espressione, è meglio usare delle parentesi in più. Troppe non faranno del male, ma troppo poche certamente causeranno dei guai.

NOTAZIONE ESPONENZIALE

I numeri sono talvolta stampati in quella che è conosciuta come notazione esponenziale o notazione E. Esempi di notazione E sono 2.14E+06 o 6.032E−07.
È facile vedere perché questa notazione particolare è necessaria sia per numeri molto grandi sia per numeri molto piccoli. Il computer di solito stampa nove cifre per numero. Il problema sorge quando vogliamo che il computer stampi una variabile il cui valore è ad esempio, 45612800000, che richiederebbe undici cifre. Il computer la stamperebbe come 4.56128E +10, che significa che il punto decimale deve andare dieci posti più a destra rispetto alla sua presente posizione. Una variabile il cui valore sia 8956000000000 sarebbe stampata come 8.956E+12. L'E+12 significa che il punto decimale deve andare dodici posti più a destra. Nello stesso modo possiamo anche rappresentare numeri piccolissimi. Una variabile il cui valore sia 0.0000000683 sarebbe presentata come 6.83E−08. L'E-08 significa che il punto decimale deve andare otto posti più a sinistra. La tavola seguente dovrebbe aiutarvi a capire come passare dalla forma decimale alla notazione E o tornare dalla notazione E alla forma decimale.

Forma decimale	Notazione E
2630000	2.63E+06
263000	2.63E+05
26300	2.63E+04
2630	2.63E+03
263	2.63E+02
26.3	2.63E+01
2.63	2.63
0.263	2.63E−01
0.0263	2.63E−02
0.00263	2.63E−03
0.000263	2.63E−04
0.0000263	2.63E−05
0.00000263	2.63E−06

Per passare dalla notazione E alla forma decimale, osservate il segno che segue la E. Se il segno è +, spostate il punto decimale a destra di tanti posti quanti sono indicati dal numero che segue il +. Se il segno dopo la E è −, spostate il punto decimale a sinistra. Per passare da un numero decimale alla notazione E, capovolgete semplicemente il procedimento. In realtà non dovreste preoccuparvi troppo della notazione E, perché la userete raramente quando imposterete i vostri programmi sul computer. Il motivo principale per cui se ne è parlato è che può capitare che il computer emetta dei numeri in notazione E. Di conseguenza, dovrete essere in grado di riconoscere quello che sta succedendo.

FORMATTARE UN DISCHETTO

La formattazione di un dischetto si esegue una sola volta nel modo seguente: si inserisce il dischetto nel drive con la tacca a sinistra e l'etichetta verso l'alto; si chiude lo sportellino del drive, si batte: OPEN 15, 8, 15, "NEW0: *nome del dischetto*, ID" e si preme RETURN. Quando cessa il ronzio si è pronti a immettere le informazioni sul dischetto. OPEN 15, 8, 15 stabilisce un canale di comunicazione (15) dal computer al drive (unità n. 8) per trasmettere l'istruzione "NEW0: *nome del dischetto*, ID". ID può essere composto da due caratteri qualsiasi.

ARCHIVIAZIONE E RICHIAMO DI PROGRAMMI

Quando si spegne il computer, si perde il programma in memoria. Se tutte le volte che accendiamo il computer dovessimo battere i programmi

che vogliamo usare, si farebbe ben poco lavoro. Fortunatamente, il C-64 vi permette di battere programmi lunghi o complicati, provarli e correggerne gli errori, e poi registrare tali programmi su un disco formattato per usi futuri. Abbiamo solo bisogno di accendere il drive, la TV e il computer, e poi ricuperare qualunque programma che si sia prima registrato. Possiamo non aver bisogno di tenere un certo programma per sempre sul disco. Esiste un comando per cancellare un programma dal disco. Se spostate una copia del programma da un disco in memoria e poi cancellate la copia che c'è sul disco, la copia che è in memoria non viene toccata. Allo stesso modo, potete avere un programma in memoria e cancellare un altro programma che si trova sul disco senza alterare il programma in memoria. Ricordate che richiamare dal disco un programma o l'elenco dei programmi cancella il programma attualmente in memoria.
I comandi per eseguire questi compiti di gestione dei programmi sono forniti qui di seguito.

Per spostare un programma dalla memoria ad un archivio su disco battere
SAVE"nome del programma",8

Per spostare un programma da un archivio su disco alla memoria battere
LOAD"nome del programma",8

Per cancellare un programma dal disco battere
OPEN15,8,15,"SCRATCH0: Nome del programma"

Per visualizzare l'elenco dei programmi sul disco battere
LOAD "$",8
LIST

Il comando SAVE non permette di registrare su programmi già memorizzati sul dischetto. Per sostituire un programma su dischetto con un programma in memoria, battete:

SAVE"@0:*nome del programma*",8

In alternativa potete cancellare (SCRATCH) il programma dal disco e usare SAVE.
Può capitare incidentalmente di perdere dei programmi memorizzati, perciò è sempre conveniente farne una copia su un altro dischetto.
A volte, usando l'istruzione OPEN, può apparire il messaggio FILE OPEN ERROR: in questo caso battete CLOSE 15 e di nuovo OPEN.
È possibile anche usare una cassetta per l'archiviazione dei programmi con il Datassette. Troverete le istruzioni dei comandi SAVE e LOAD sul

manuale del C-64. L'uso delle cassette è più lento e meno affidabile di quello dei dischetti.

3.4 Test di apprendimento

1. Scrivere i simboli che sono usati per eseguire le seguenti operazioni aritmetiche nelle espressioni BASIC:

   a. Moltiplicazione

   ....

   b. Elevazione a potenza

   ....

   c. Divisione

   ....

2. Quando le espressioni aritmetiche vengono calcolate, esiste una priorità delle operazioni. Qual è questa priorità?

   a. 1°

   ....

   b. 2°

   ....

   c. 3°

   ....

3. Quale operazione aritmetica è eseguita per prima in questa istruzione in modo diretto?

```
PRINT(3+5)↑2
```

....

4. Quando esplora le espressioni aritmetiche il computer ricerca in una direzione specifica. Qual è questa direzione?

....

5. Scrivete un'istruzione BASIC per calcolare la seguente espressione. Date alla riga il numero 100.

$$A = (4+3B/D)^2$$

....

6. Se il seguente programma venisse eseguito, che cosa sarebbe stampato?

```
100 LET A=2
110 LET B=3
120 LET C=(A*B+2)/2
130 PRINT C
140 END
```

....

7. Convertite nella notazione E i seguenti numeri.

   a. 5160000000

   ....

   b. 0.0000314

   ....

8. Convertite in forma decimale i seguenti numeri.

   a. 7.258E+06

   ....

   b. 1.437E−03

   ....

9. Date l'ordine in cui le operazioni nell'espressione seguente verranno eseguite dal computer.

```
100 LET A=(6/3+4)↑2
```

....

10. Quali comandi sono necessari per eseguire le seguenti operazioni:

    a. Spostare un programma dal disco alla memoria.

    ....

    b. Spostare un programma dalla memoria al disco.

    ....

    c. Cancellare un programma dal disco.

    ....

    d. Cancellare un programma dalla memoria.

    ....

    e. Visualizzare il programma in memoria.

    ....

    f. Eseguire il programma in memoria.

    ....

    g. Visualizzare i nomi di tutti i file archiviati sul disco.

    ....

11. Supponete di star battendo una riga di programma sul computer e di non aver ancora premuto RETURN. Come si corregge un unico carattere?

    ....

Capitolo

Input, output e semplici applicazioni

4.1 Obiettivi

In questo capitolo vedremo in pratica come si scrivono i programmi. Aumenteremo anche la nostra conoscenza del BASIC osservando alcuni dettagli circa l'ingresso (input) e l'uscita (output) dei dati. Gli obiettivi sono i seguenti:

IMMISSIONE DI NUMERI IN UN PROGRAMMA BASIC

Vi sono tre modi di immettere numeri nel computer per un programma BASIC. Dal momento che il computer tratta principalmente numeri, è necessario che sappiamo come immetterli.

STAMPA DI VARIABILI E STRINGHE

Dopo che le informazioni sono state calcolate, devono essere presentate. Di solito vogliamo avere in uscita sia stringhe di caratteri, che numeri. L'uscita delle stringhe è trattata essenzialmente allo stesso modo dell'uscita dei numeri, ma richiede una speciale attenzione.

SPAZIATURA DELL'OUTPUT

Il computer mette gli spazi in uscita in maniera standard. Imparerete come modificare questi spazi.

L'ISTRUZIONE REM

Il programmatore prudente inserisce dei commenti nei suoi programmi come aiuto per spiegare o interpretare quello che si sta facendo. L'istruzione REM in BASIC ci permette di fare questo.

ESEMPI DI PROGRAMMI

Il nostro scopo ultimo è quello di imparare a scrivere e controllare i programmi, correggendone gli eventuali errori. In questo capitolo cominceremo con qualche compito piuttosto facile.

4.2 Esercizi di scoperta

1. Accendete il computer e la TV. Immettete il seguente programma:

```
100 INPUT A
110 INPUT B
120 INPUT C
130 LET D=A+B+C
140 PRINT D
150 END
```

   Che cosa pensate che avverrà quando eseguite questo programma?

   ....

   Eseguite il programma. Quando compare il primo punto interrogativo (il segnale dell'input per A), battete 2. Quando compare il secondo punto interrogativo, battete 3, e infine, all'ultimo punto interrogativo, battete 5. Registrate qui sotto quello che succede.

   ....

2. Si noti come nel programma al punto 1 abbiamo tre istruzioni di ingresso (INPUT), alle righe 100, 110 e 120. Battete

```
100
110
```

Che cosa succede al programma?

....

Visualizzate il programma in memoria e controllate se avevate ragione. Quindi battete

```
120 INPUT A,B,C
```

Visualizzate il programma. Che cosa è successo?

....

3. Eseguite il programma e quando compare il segnale di input (?), battete

```
2,3,5
```

Che cosa è successo?

....

Si può immettere in ingresso più di una variabile alla volta in un programma BASIC?

....

4. Eseguite il programma e quando compare il segnale di input, battete

```
2,3
```

Che cosa è successo?

....

Che cosa sta aspettando il computer?

....

Battete

```
2,3,5
```

e segnate qui sotto quello che succede.

....

Notate che il 2 in 2, 3, 5 è usato come terzo input mentre il 3 e il 5 vengono ignorati.

....

5. Eseguite il programma e quando il segnale di input compare, battete

```
2,3,5,1
```

Che cosa è successo?

....

6. Potete immettere più numeri di quanti sono richiesti da un'istruzione INPUT?

....

Che cosa succede se lo fate?

....

7. Potete immettere meno numeri di quanti sono richiesti da un'istruzione INPUT?

....

Che cosa succede se lo fate?

....

8. Battete

```
120 READ A,B,C
```

Visualizzate il programma. Che cosa è successo?

....

Eseguite il programma e registrate quello che succede.

....

9. Ora battete

```
125 DATA 2,3,5
```

e visualizzate il programma. Che cosa è successo?

....

10. Eseguite il programma e registrate qui sotto quello che succede.

....

Da quello che avete appena visto, potete desumere che tutte le volte che un programma BASIC contiene un'istruzione READ ("leggi"), dev'esserci nel programma un altro tipo di istruzione. Qual è questa istruzione?

....

11. Indicate due diversi metodi (all'infuori di un'istruzione LET) per immettere numeri in un programma. (Suggerimento: vedere i punti 2 e 8).

....

12. Visualizzate il programma in memoria. Cancellate l'istruzione DATA. Battete

```
145 DATA 2,3,5
```

e visualizzate di nuovo il programma. Che cosa è successo?

13. Eseguite il programma e segnate qui sotto l'uscita.

....

Secondo voi, cambia qualcosa se si modifica il posto dell'istruzione DATA nel programma?

....

14. Cancellate il programma in memoria. Immettete il seguente

```
100 READ A,B
110 LET C=A/B
120 PRINT C
130 GOTO 100
140 DATA 2,1,6,2,90,9,35,7
150 END
```

Che cosa pensate che succeda se eseguite il programma?

....

Provatelo e vedete un po' se avevate ragione. Scrivete l'uscita.

....

Il messaggio OUT OF DATA (non ci sono più dati) è associato all'istruzione READ o all'istruzione DATA?

....

15. Cancellate l'istruzione DATA nella riga 140 del programma. Ora immettete le seguenti istruzioni:

```
105 DATA 10,2
115 DATA 100,50
125 DATA 50,5
```

Visualizzate il programma in memoria. Che cosa è successo?

....

16. Se si esegue il programma, che cosa pensate che sarà visualizzato?

....

Eseguite il programma e vedete se avevate ragione. Scrivete qui sotto l'uscita.

....

17. Si può avere più di un'istruzione DATA in un programma BASIC?

....

Vi sembra che la posizione delle istruzioni DATA nel programma abbia qualche influenza?

....

18. Cancellate il programma in memoria. Immettete il programma seguente

```
100 LET A=10
110 PRINT A
120 END
```

Che cosa succederà se si avvia il programma?

....

Eseguite il programma e registrate qui sotto che cosa è successo.

....

19. Ora battete

```
110 PRINT "A"
```

e visualizzate il programma in memoria. Che cosa è successo?

....

Che cosa succederà se si esegue il programma?

....

Avviate il programma e registrate quello che il computer ha visualizzato.

....

20. Battete

```
110 PRINT "CANE DA CACCIA = ";A
```

e visualizzate il programma in memoria. Che cosa pensate che accadrà se eseguite il programma?

....

Avviate il programma e scrivete quello che è successo.

....

21. Ora proviamo un aspetto diverso. Battete

```
110 PRINT "B = ";A
```

Visualizzate il programma e studiatelo attentamente. Se il programma viene avviato, che cosa pensate che succederà?

....

Provatelo e vedete se avevate ragione. Registrate qui sotto l'uscita.

....

22. La riga seguente è più lunga dei 40 spazi disponibili su una riga di schermo. Non premete RETURN quando arrivate alla fine dello schermo, e continuate a battere:

```
90 REM QUESTO E' UN PROGRAMMA DIMOSTRATIVO
```

Visualizzate il programma. Che cosa è successo?

....

Le righe di programma non devono essere più lunghe di 2 righe di schermo (80 caratteri). Eseguite il programma. Com'è l'uscita?

....

L'istruzione REM della riga 90 ha qualche effetto sul programma?

....

23. Cancellate il programma in memoria. Immettete il seguente programma:

```
100 REM PROGRAMMA DI CONVERSIONE METRICA
110 REM CONVERTE LE LIBBRE IN GRAMMI
120 PRINT "IMMETTERE N.DI LIBBRE";
130 INPUT P
140 LET G=454*P
```

```
150 PRINT P;"LIBBRE SONO";G;"GRAMMI"
160 GOTO 120
170 END
```

Visualizzate il programma e verificate che sia giusto. Studiatelo attentamente e cercate di pensare quello che accadrà se fosse eseguito. Ora avviate il programma. Quando il messaggio di ingresso viene visualizzato, immettete un numero a piacere. Notate quello che viene presentato in risposta. Ripetete varie volte questo procedimento, poi fate uscire il computer dall'iterazione di ingresso. Se avete dimenticato come si fa, vedete il Capitolo 2 al punto 22 degli Esercizi di scoperta. Qual è lo scopo dell'istruzione REM?

....

24. Battete

```
115 INPUT P
130
160 GOTO 115
```

e quindi visualizzate il programma in memoria. Che cosa è successo?

....

Il programma funzionerebbe in questa forma?

....

Eseguite il programma e, al messaggio di input, battete 1. Che cosa è successo?

....

Fate uscire il programma dall'iterazione di input.

25. Facciamo ancora qualche altro esperimento con questo programma. Cancellate il programma dalla memoria e immettetelo di nuovo, modificato come segue:

```
100 REM PROGRAMMA DI CONVERSIONE METRICA
110 REM CONVERTE LE LIBBRE IN GRAMMI
120 PRINT "IMMETTERE N.DI LIBBRE ";
130 INPUT P
```

```
140 PRINT P;" LIBBRE SONO ";G;" GRAMMI"
150 LET G=454*P
160 GOTO 120
170 END
```

Il programma può essere eseguito in questa forma?

....

Avviate il programma e, al messaggio di input, battete 2. Che cosa è successo?

....

Spiegate con parole vostre che cosa c'è che non va. Ricordate che, se una variabile non viene inizialmente definita nel programma, il computer la porrà uguale a zero.

....

Fate uscire il programma dall'iterazione di input.

26. Cancellate il programma dalla memoria. Immettete il seguente:

```
100 READ A
110 PRINT A
120 GOTO 100
130 DATA 10,12,8,9,11223344556б,-82,5,6
140 END
```

Eseguite il programma e registrate quello che avviene.

....

27. Battete

```
110 PRINT A,
```

Notate che tutto quello che abbiamo fatto è stato di inserire una virgola dopo la A nella 110. Eseguite il programma. Sono allineati i numeri sulle ultime due colonne?

....

28. Ora sostituite la virgola dopo la A con un punto e virgola battendo

```
110 PRINT A;
```

Eseguite il programma e registrate quanto avviene.

....

29. Se una variabile in un'istruzione PRINT non è seguita da alcun segno di interpunzione, come mette gli spazi in uscita il computer? (Suggerimento: vedere il punto 26).

....

E se la variabile è seguita da una virgola?

....

Che cosa succede se la variabile è seguita da un punto e virgola?

....

30. Cancellate il programma dalla memoria ed immettete il programma seguente:

```
100 LET A=15
110 READ B
120 PRINT TAB(A);B;
130 LET A=A+10
140 GOTO 110
150 DATA 1,2,3
160 END
```

Eseguite il programma e registrate quello che avviene.

....

31. Battete

```
130 LET A=A+5
```

Eseguite il programma e scrivete quello che è successo.

....

32. Battete

```
130 LET A=A+20
```

Eseguite il programma e segnate quello che è successo.

....

33. Che cosa sembra controllare il **TAB** nell'istruzione di stampa?

....

34. Con questo si conclude per ora il lavoro sul computer. Spegnete il computer e la TV.

4.3 Analisi

In questo capitolo abbiamo cominciato a staccarci dalla pura meccanica del controllo del computer. Ci concentriamo ora invece sulla scrittura e sul controllo dei programmi. Quest'abilità non viene naturale alla maggior parte delle persone e per questo riserveremo all'argomento molta attenzione, sia adesso, che nei capitoli seguenti.

IMMISSIONE DI NUMERI IN UN PROGRAMMA BASIC

Nel Capitolo 2 abbiamo visto uno dei modi per immettere i numeri nei programmi, assegnando cioè dei valori ad una variabile nel programma stesso. Per esempio

```
100 LET A=6
```

introduce il valore 6 in un programma e memorizza il numero sotto il nome di variabile A. Questo metodo ha dei limiti. Abbiamo bisogno di esaminare altri modi in cui i numeri possono essere introdotti in un programma BASIC. Vediamo prima l'istruzione INPUT e come viene usata. Un esempio potrebbe essere

```
260 INPUT G
```

Quando il computer esegue questa riga, visualizzerà un punto interrogativo per segnalare che aspetta un ingresso di dati, quindi si ferma e

aspetta che battiate il numero e RETURN. Nel caso suddetto il numero battuto sarà conosciuto come G.
In un'unica istruzione INPUT può essere richiamata più di una variabile, come

```
420 INPUT A,B,C,D
```

In questo caso viene presentato lo stesso segnale di input (il punto interrogativo), ma ora il computer si aspetta che siano battuti quattro numeri separati da virgole. Se sono immessi solo tre numeri (o meno) e viene premuto il tasto RETURN il computer risponderà con "??". Se vengono immessi inizialmente più di quattro numeri, comparirà il messaggio: EXTRA IGNORED.
L'ultimo metodo per fornire numeri in ingresso al computer è con le istruzioni READ e DATA. L'istruzione

```
100 READ A,B,C,D
```

è trattata dal computer allo stesso modo dell'istruzione INPUT, ma con due differenze. Primo, il computer non si ferma. Non ce n'è bisogno, come vedremo. Secondo, i numeri richiesti sono letti nelle istruzioni DATA contenute all'interno del programma, invece di essere immessi al terminale in risposta ad un segnale di input.
Il programma seguente illustra le istruzioni READ e DATA.

```
100 READ A,B,C,D
110 LET E=A+B+C+D
120 PRINT E
130 DATA 25,3,17,12
140 END
```

Il programma legge quattro numeri dalla riga DATA e stampa la somma dei numeri. Non fa differenza dove si trova l'istruzione DATA nel programma. Ci può essere più di una istruzione DATA, e queste non hanno bisogno di essere raggruppate assieme nello stesso posto nel programma. Man mano che i numeri sono richiamati da istruzioni READ, vengono presi in ordine nelle istruzioni DATA, a partire dalla riga DATA con il numero più basso. Se l'istruzione READ richiede altri numeri dopo che sono stati usati tutti quelli disponibili nelle istruzioni DATA, il computer emetterà un messaggio OUT OF DATA e quindi si fermerà.
Per riassumere, ci sono tre metodi per introdurre numeri nei programmi BASIC. Usare 1) le istruzioni LET, 2) le istruzioni INPUT, 3) le istruzioni READ e DATA. Ci sono occasioni in cui ciascuno di questi tre metodi può essere usato con profitto. Vi familiarizzerete con i vantaggi e gli svantag-

gi di ciascun metodo man mano che dedicherete più tempo alla programmazione.

Si possono immettere numeri in un programma BASIC con:
1) LET
2) INPUT
3) READ/DATA

STAMPA DI VARIABILI E STRINGHE

Il computer può emettere o il valore numerico di una variabile (un numero) o una stringa di caratteri. Per illustrare ciò, supponiamo di avere una variabile chiamata X e che il numero 2 sia memorizzato in quella posizione di memoria. Il programma

```
100 LET X=2
110 PRINT "X"
120 PRINT X
130 END
```

mostra la differenza fra l'uscita di una stringa e quella di una variabile. La riga 110 emette il carattere X perché X è racchiuso fra virgolette. La riga 120 stampa 2 perché quello è il numero memorizzato nella posizione di memoria X.
La regola è chiara. I caratteri contenuti fra virgolette sono chiamati stringhe. Le stringhe vengono emesse esattamente come sono elencate. Il computer non tenta di analizzare o di scoprire quello che c'è in una stringa. Se una variabile in un'istruzione PRINT non è contenuta fra virgolette, il computer stampa il valore numerico di quella variabile.
È possibile fare calcoli in un'istruzione PRINT. Così

```
100 PRINT A+B+C,D
```

farà sì che il computer emetta la somma dei numeri memorizzati in A, B e C, e poi il numero memorizzato in D. Naturalmente le variabili A, B, C e D devono essere state prima definite, altrimenti il computer assegnerà loro valore zero.

SPAZIATURA DELL'OUTPUT

Il BASIC C-64 ha un meccanismo di spaziatura "incorporato" che stampa i numeri in posti prefissati sulla riga. Quando le quantità in un'istruzione

PRINT sono separate da virgole il computer usa la spaziatura standard. Le quattro posizioni standard di stampa sono 0, 10, 20 e 30. Il primo spazio di ciascun numero è quello per il segno −, se è necessario. La virgola segnala al computer di spostarsi alla successiva posizione di stampa sulla riga. Se il computer si trova già nell'ultima posizione sulla riga e si imbatte in una virgola in un'istruzione **PRINT**, allora effettua un ritorno carrello e stampa il numero della prima posizione della riga successiva. Così se **A**, B, C, D e E sono numeri interi

```
100 PRINT A,B,C,D,E
```

farà sì che i valori numerici di A, B, C, D, siano stampati spaziati uniformemente lungo una riga nelle 4 posizioni standard. Il valore numerico di E sarebbe stampato sotto il valore di A sulla riga successiva.
Se A, B, C, D e E sono numeri che richiedono una notazione esponenziale lunga, A e B saranno stampati sulla prima riga, C e D sulla seconda nelle due posizioni standard, mentre E sarà stampato sulla 3ª riga sotto A e C.

Le virgole nelle istruzioni PRINT producono colonne allineate

Il punto e virgola produce una spaziatura più ristretta della spaziatura standard. Per esempio l'uscita di

```
100 PRINT A;B;C;D;E
```

produce una spaziatura minore rispetto a un'istruzione PRINT con virgole. Il punto e virgola fa lasciare al computer uno spazio per i numeri negativi e due per quelli positivi. Due o più numeri possono essere messi su una riga a seconda della loro misura e formato.
Infine, possiamo controllare la spaziatura su una riga più precisamente con la funzione TAB nelle istruzioni PRINT. La funzione TAB lavora allo stesso modo del posizionamento del tabulatore su una macchina da scrivere. Sullo schermo ci sono quaranta posizioni disponibili per il tabulatore (0-39), cominciando alla posizione TAB 0.
L'istruzione

```
100 PRINT TAB(5);A;TAB(25);B
```

segnala al computer di spaziare fino alla 5ª posizione di stampa, stampare il valore numerico di A (nella 6ª), spaziare fino alla venticinquesima posizione, e infine stampare il valore numerico di B. Se B ha troppe cifre da inserire sulla riga, il calcolatore metterà B all'inizio della linea successiva, se c'è una virgola prima di B. Se B è preceduto da un ; le cifre ulteriori saranno messe sulla riga successiva.
È anche possibile avere un posizionamento variabile del tabulatore, controllato dal computer:

```
100 PRINT TAB(X);A
```

Qui il computer deve prima cercare il valore di X, quindi spaziare fino alla posizione di stampa determinata dall'intero più grande in X (per esempio, se X = 23.1435, il computer spazierà fino alla ventitreesima posizione di stampa), quindi metterà il valore numerico di A nella ventiquattresima posizione di stampa.

Usare la funzione TAB per produrre spaziature variabili

Un ultimo commento circa l'istruzione PRINT. Possiamo produrre una spaziatura verticale nell'uscita usando un'istruzione PRINT vuota, come segue:

```
100 PRINT
```

Il computer va a cercare la quantità che dev'essere stampata e non ne trova. Cerca quindi la punteggiatura e, non trovandone alcuna, ordina un ritorno carrello e fa "avanzare la carta" di una riga. Se volessimo due o tre righe vuote in uscita, potremmo ottenere la spaziatura usando due o tre istruzioni PRINT vuote.

L'ISTRUZIONE REM

L'istruzione REM (REMark significa "commento") è molto diversa dalle istruzioni che abbiamo visto prima. Non appena il computer legge i caratteri REM dopo il numero di riga, ignora il resto dell'istruzione e passa alla riga successiva. L'istruzione REM è un modo di fornire informazioni a beneficio del programmatore o di qualcuno che legga il programma. Queste informazioni rendono molto più facile seguire quello che sta succedendo nel programma.
Per illustrare l'uso delle istruzioni REM esaminiamo due programmi. Entrambi produrranno risultati identici, ma il secondo usa delle istruzioni REM per descrivere quello che succede nel programma. Potete giudicare da voi stessi quale dei due programmi sia più facile da capire.

1. Nessuna istruzione REM:

```
100 INPUT A,B,C,D
110 LET X=(A+B+C+D)/4
120 PRINT X
130 END
```

2. Con istruzioni REM:

```
100 REM CALCOLA LA MEDIA DI QUATTRO NUMERI
110 REM INSERISCE I QUATTRO NUMERI
120 INPUT A,B,C,D
130 REM CALCOLA LA MEDIA
140 LET X=(A+B+C+D)/4
150 REM STAMPA LA MEDIA
160 PRINT X
170 END
```

4.4 Esempi di programmi

Nei prossimi capitoli passeremo progressivamente più tempo scrivendo e provando i programmi. Gli esempi scelti per questo capitolo sono molto semplici, ma illustrano le idee che abbiamo esposto. Studiate ogni esempio attentamente fino a quando siete certi di averne capito tutti i dettagli. Non sarà male immettere i programmi nel computer ed eseguirli per verificare che funzionino a dovere.

ESEMPIO 1 — PREZZI UNITARI

Il nostro problema è quello di scrivere un programma che calcoli i prezzi unitari delle merci di un supermercato. Chiameremo T il prezzo totale, N il numero dei pezzi e U il prezzo unitario. Possiamo calcolare il prezzo unitario come segue.

$$U = T/N$$

Per esempio, supponiamo il caso che dodici lattine di succo di frutta costino 6960 lire. Il costo unitario per lattina sarà

$$U = 6960/12 = 580 \text{ lire}$$

Vogliamo che il programma sia ideato in modo che quando venga eseguito produca l'uscita seguente:

QUAL È IL PREZZO TOTALE? (Si immette il valore di T)
QUAL È IL NUMERO DEI PEZZI? (Si immette il valore di N)
IL PREZZO UNITARIO È (Il computer visualizza il valore di U)

Ora esamineremo questo esempio per vedere come il programma è collegato con quello che vogliamo vedere in uscita.

QUAL È IL PREZZO TOTALE? (Immissione di T)
100 200
QUAL È IL NUMERO DEI PEZZI? (Immissione di N)
300 400
500: (Calcola il prezzo unitario)
IL PREZZO UNITARIO È (Uscita di U)
600
700: (Fine del programma)

Scriveremo ciascuna riga del programma in modo che i numeri sotto le varie parti siano i numeri di riga del programma. Nella riga 100 vogliamo che il computer visualizzi il messaggio indicato. Questo si ottiene con il comando PRINT.

```
100 PRINT "QUAL E' IL PREZZO TOTALE ";
```

Si noti il punto e virgola fuori delle virgolette. Il motivo di ciò è che non vogliamo un ritorno carrello, ma vogliamo che la riga stampata stia lì in attesa del segnale dell'input. La riga 200 dovrebbe essere un'istruzione INPUT per chiedere l'immissione di T.

```
200 INPUT T
```

Il messaggio della riga 300 richiede un'istruzione PRINT.

```
300 PRINT "QUAL E' IL NUMERO DEI PEZZI ";
```

L'immissione del numero totale dei pezzi è trattata allo stesso modo del prezzo totale.

```
400 INPUT N
```

Poi si calcola il prezzo unitario nella riga 500.

```
500 LET U=T/N
```

La riga successiva è un messaggio seguito dal prezzo unitario, ed è risolta con un'istruzione PRINT.

```
600 PRINT "IL PREZZO UNITARIO E' ";U
```

Infine abbiamo l'istruzione END.

```
700 END
```

Ora mettiamo insieme tutto il programma.

```
100 PRINT "QUAL E' IL PREZZO TOTALE ";
200 INPUT T
300 PRINT "QUAL E' IL NUMERO DEI PEZZI ";
400 INPUT N
500 LET U=T/N
600 PRINT "IL PREZZO UNITARIO E' ";U
700 END
```

ESEMPIO 2 — CONVERSIONE DELLA TEMPERATURA

La relazione fra la temperatura misurata in gradi Fahrenheit e in gradi Celsius è

$$C = (5/9)(F-32)$$

dove C sta per gradi Celsius e F sta per gradi Fahrenheit. Se, per esempio, F è 212, allora C sarà

$$C = (5/9)(212-32) = 100$$

Come nel primo esempio, scriveremo il programma dopo aver visto come vogliamo che appaia l'uscita.

IMMETTERE N. GRADI F
? (Si immette il valore di F)
(Valore di F) GRADI F EQUIVALE A (Risposta) GRADI C

Ancora una volta divideremo l'uscita in parti che saranno generate dalle righe del programma:

IMMETTERE N. GRADI F
100
200: Immissione di F
300: Calcola i gradi C
(Uscita di F) GRADI F EQUIVALE A (Uscita di C) GRADI C
400
500: (Fine del programma)

Il programma corrispondente è

```
100 PRINT "IMMETTERE N. GRADI F "
200 INPUT F
300 LET C=(5/9)*(F-32)
400 PRINT F;" GRADI F EQUIVALE A ";C;" GRADI C"
500 END
```

Questo programma è un po' diverso dal precedente. Nella riga 100 non vi è punteggiatura dopo la stringa. Così il segnale di input generato dall'istruzione INPUT nella riga 200 sarà visualizzato nella riga che segue la stringa iniziale. L'istruzione PRINT della riga 400 emette: 1) il valore di F, 2) una stringa, 3) il valore di C, 4) una seconda stringa. I punti e virgola sono usati nella riga 400 per collegare le variabili e le stringhe nell'istruzione PRINT.

ESEMPIO 3 — SOMMA E PRODOTTO DI NUMERI

L'esempio finale di questo capitolo riguarda il calcolo della somma e del prodotto di due numeri. Ancora una volta scriveremo il programma per ottenere l'uscita desiderata.

INPUT A? (Si immette il valore di A)
INPUT B? (Si immette il valore di B)
LA SOMMA DI A E B È (Il computer visualizza la somma)
IL PRODOTTO DI A E B È (Il computer visualizza il prodotto)
(Vengono inserite dal computer delle righe vuote)
INPUT A? (Si immette il valore di A)
(ecc.)

Con questo problema procederemo più rapidamente. La prima riga dell'uscita desiderata può essere risolta con le due istruzioni seguenti:

```
100 PRINT "INPUT A ";
110 INPUT A
```

Si noti attentamente che il messaggio stampato nella riga 100 è solo di aiuto per l'utente; l'istruzione di immissione importante per il computer è nella riga 110.
Generiamo la seconda riga dell'uscita desiderata come segue:

```
120 PRINT "INPUT B ";
130 INPUT B
```

La somma e il prodotto di due numeri sono emessi nel modo seguente:

```
140 PRINT "LA SOMMA DI A E B E' ";A+B
150 PRINT "IL PRODOTTO DI A E B E' ";A*B
```

Il programma calcola la somma e il prodotto di 2 numeri e si ferma. Potete far sì che il programma continui a calcolare aggiungendo un loop (GOTO); potete anche separare le uscite includendo le righe vuote generate da istruzioni PRINT vuote, come segue:

```
160 PRINT
170 PRINT
180 GOTO 100
```

Naturalmente, la riga finale dovrà essere l'istruzione END.

```
190 END
```

L'intero programma è listato qui sotto.

```
100 PRINT "INPUT A ";
110 INPUT A
120 PRINT "INPUT B ";
130 INPUT B
140 PRINT "LA SOMMA DI A E B E' ";A+B
150 PRINT "IL PRODOTTO DI A E B E' ";A*B
160 PRINT
170 PRINT
180 GOTO 100
190 END
```

Così com'è stato strutturato, il programma continuerà a tornare sui propri passi fino a che non lo facciamo uscire dall'iterazione di input. Po-

tremmo calcolare la somma e il prodotto dei due numeri in righe separate, come nella seguente versione dello stesso programma.

```
100 PRINT "INPUT A ";
110 INPUT A
120 PRINT "INPUT B ";
130 INPUT B
140 LET S=A+B
150 PRINT "LA SOMMA DI A E B E' ";S
160 LET P=A*B
170 PRINT "IL PRODOTTO DI A E B E' ";P
180 PRINT
190 PRINT
200 GOTO 100
210 END
```

Un'ultima osservazione: studiate i dettagli di un programma con carta e matita, prima di lavorare sul computer. Risparmierete tempo e fatica.

4.5 Problemi

1. Scrivere un programma che legga i quattro numeri 10, 9, 1 e 2 da un'istruzione DATA e metta i numeri rispettivamente in A, B, C e D. Sommare i primi due numeri mettendo la somma in S. Quindi calcolare il prodotto degli ultimi due numeri, mettendo il risultato in P. Stampare il valore di S e di P sulla stessa riga.

2. Scrivere un programma che richieda l'immissione di quattro numeri, quindi stampi i numeri in ordine inverso. Per esempio, se battete 5, 2, 11, 12, il computer deve stampare in risposta 12, 11, 2, 5. Il programma deve funzionare per qualunque gruppo di numeri che si decida di immettere. Usate nel vostro programma solo 4 righe, compresa l'istruzione END.

3. Scrivere un programma che legga le variabili A, B, C e D da numeri di vostra scelta in un'istruzione DATA. Stampare i numeri verticalmente con B sotto A, C sotto B, e D sotto C.

4. Che cosa sarà emesso in uscita se viene eseguito il seguente programma?

```
100 READ X,Y,Z
110 DATA 2,5,3
```

```
120 LET T=X*Y+Z
130 LET S=Y↑2
140 PRINT T,S
150 END
```

5. Che cosa c'è che non va in questo programma?

```
100 LET A=2
110 READ B
120 LET A=A+C/B
130 DATA 3
140 PRINT A
150 END
```

6. Spiegate con parole vostre quello che fa il programma seguente.

```
100 INPUT A,B
110 LET S=A+B
120 LET T=A-B
130 LET U=A*B
140 PRINT S;T;U
150 END
```

7. Uno degli indici usati per giudicare lo stato di salute di un'impresa è il rapporto tra il totale del contante disponibile più i titoli commerciabili e i crediti, diviso per i debiti correnti. Scrivere un programma che richieda l'immissione delle quantità necessarie, quindi calcoli l'indice richiesto.

8. Scrivere un programma che conti e stampi per cinque a cominciare da 0. I primi numeri dovranno essere 0, 5, 10, 15, e così via. Interrompere manualmente il programma quando sono stati emessi quaranta o cinquanta numeri.

9. L'uscita del programma riportato qui sotto dovrebbe essere 1, 3, 5, 7, 9 e così via, ma il programma contiene un errore. Che cosa c'è che non va?

```
100 LET A=1
110 PRINT A;
120 LET A=A+2
130 GOTO 100
140 END
```

10. Se un oggetto è lasciato cadere vicino alla superficie della terra, la

distanza che percorrerà in caduta in un determinato tempo sarà determinata da

M = 9.8S↑2

dove M è la distanza percorsa in caduta (in metri) e S è il tempo della caduta in secondi. Scrivere un programma che, quando lo si esegue, produca la seguente uscita:

TEMPO DI CADUTA (SEC)? (Immettere S)
L'OGGETTO CADE PER (Valore di M) METRI

11. Il volume di una scatola può essere calcolato come

V = LPA

dove L, P, A sono rispettivamente la larghezza, la profondità e l'altezza. Se queste vengono tutte misurate in centimetri, per esempio, il volume sarà in centimetri cubi. Vogliamo un programma che, quando avviato, dia l'uscita seguente:

LARGHEZZA (CM)? (Si immette L)
PROFONDITÀ (CM)? (Si immette P)
ALTEZZA (CM)? (Si immette A)
IL VOLUME È (Il computer emette V) CM CUBI

Il programma seguente è sbagliato e non darà l'uscita richiesta qui sopra. Che cosa c'è che non va?

```
100 PRINT "LARGHEZZA (CM)";L
110 PRINT "PROFONDITA' (CM)";P
120 PRINT "ALTEZZA (CM)";A
130 INPUT L,P,A
140 LET V=L*P*A
150 PRINT "IL VOLUME E'";V;"CM CUBI "
160 END
```

12. Nel programma sottostante due numeri, A e B, sono richiesti nell'istruzione di ingresso. Il problema è di fornire le istruzioni mancanti in modo che, quando A e B vengono stampati, i loro valori siano stati scambiati fra loro.

```
100 INPUT A,B
110
120
```

```
130
140 PRINT A,B
150 END
```

13. Supponiamo che il contachilometri della vostra macchina segni R_1 chilometri quando il serbatoio è pieno. Guidate fino a quando il contachilometri segna R_2. A questo punto sono necessari G litri di benzina per riempire il serbatoio. I chilometri per litro che avete consumato nel viaggio sono

$$K = (R_2 - R_1)/G$$

Scrivere un programma per ricavare il consumo con i seguenti dati:

R_1	R_2	G
21423	21493	5
05270	05504	13
65214	65559	11.5

14. Supponiamo che stiate scrivendo un programma per stampare dei numeri su un modulo. Si presuma che ci siano tre numeri da stampare e che questi si trovino in un'istruzione DATA. Scrivete un programma che faccia stampare il primo numero 20 caratteri in dentro dal margine sinistro del dispositivo di uscita, seguito da due righe vuote. Stampate il secondo numero cominciando al carattere 10. Stampate una riga vuota seguita da un terzo numero che cominci al carattere 15. Mettete qualunque numero vogliate nell'istruzione DATA e provate il programma.

15. Si sa che un'istruzione DATA contiene i voti d'esame di una classe di dieci studenti. Scrivete un programma che contenga non più di quattro istruzioni contando le istruzioni DATA e END per calcolare e stampare la media della classe. Provate il programma su dei dati di prova di vostra scelta.

16. C'è una vecchia storia a proposito di un saggio che avrebbe inventato gli scacchi e avrebbe chiesto come ricompensa di ricevere un chicco di grano sulla prima casella della scacchiera, 2 chicchi sulla seconda, 4 sulla terza, 8 sulla quarta, e così via. Scrivete un programma che stampi il numero della casella e il numero dei chicchi di grano su quella casella. Il programma dovrebbe contenere un'iterazione che usa un'istruzione GOTO e dovrebbe essere interrotto da tastiera

quando avete visto abbastanza. Quanti grani di frumento ci saranno sulla sessantaquattresima casella? Avviate il programma e scopritelo.

17. Se viene pagato l'interesse composto, il tasso reale annuo di interesse è più alto del tasso nominale stabilito per l'investimento. La formula BASIC seguente calcola questo tasso reale annuo di interesse:

    T = ((1+R/(100*M))↑M−1)*100

    In questa espressione, T è il tasso reale annuo di interesse in percentuale, R è l'interesse nominale in percentuale, e M è il numero di volte che l'investimento è composto per anno. Scrivere un programma che produca l'uscita seguente:

    TASSO DI INTERESSE FISSATO (%)
    ? (Introdurlo da tastiera)
    QUANTE VOLTE È COMPOSTO ALL'ANNO
    ? (Introdurlo da tastiera)
    IL TASSO REALE ANNUO DI INTERESSE È
    (Il computer emette la risposta)

18. L'interesse semplice su un investimento è calcolato secondo la regola seguente:

    I = (P)(R/100)(T/365)

    dove P è il capitale investito al tasso R di interesse annuo (espresso in percentuale) per un tempo T (espresso in giorni). Scrivere un programma che generi sul video l'uscita presentata qui sotto quando viene eseguito.

    QUAL È IL CAPITALE?
    ? (Immettere da tastiera il capitale)
    QUAL È IL TASSO ANNUO DI INTERESSE (%)?
    ? (Immettere da tastiera il tasso di interesse)
    QUAL È IL PERIODO IN GIORNI?
    ? (Immettere da tastiera il periodo)

    PER UN INVESTIMENTO DI
    (Il computer scrive il capitale)
    AD UN TASSO DI INTERESSE ANNUO DEL
    (Il computer scrive il tasso) %
    INVESTITO PER
    (Il computer scrive il periodo)

GIORNI, L'INTERESSE È
(Il computer scrive l'interesse)

19. Se una somma di denaro **P** è lasciata ad accumulare interessi al tasso di I per cento all'anno per N anni, il denaro crescerà fino all'importo totale T dato da

    T = P*(1+I/100)↑N

    Ad esempio, se **P** = 1000000, I = 6 per cento, e N = 5 anni,

    T = 1000000*(1+6/100)↑5 = 13382300

    Scrivere un programma che, quando avviato, generi la seguente uscita:

    INVESTIMENTO INIZIALE? (Si immette P)
    TASSO DI INTERESSE ANNUO (%)? (Si immette I)
    ANNI DI ACCUMULO INTERESSI? (Si immette N)
    IL VALORE TOTALE È (Il computer scrive T)

20. Se una somma di denaro **P** è lasciata ad accumulare interessi al tasso I per cento composto J volte all'anno per N anni, il valore dell'investimento sarà

    T = P*(1+I/(100*J))↑(J*N)

    Scrivere un programma che richieda l'immissione di P, I, J e N. Eseguire il programma come richiesto per ottenere il valore di 1000000 investito per due anni all'8 per cento composto

    a. Annualmente (J = 1)
    b. Semestralmente (J = 2)
    c. Mensilmente (J = 12)
    d. Settimanalmente (J = 52)
    e. Giornalmente (J = 365)

4.6 Test di apprendimento

1. Quale sarebbe l'uscita se questo programma venisse eseguito?

```
100 LET X=1
110 PRINT X;
120 LET X=X+1
130 GOTO 110
140 END
```

....

2. Descrivete tre modi di introdurre i numeri in un programma BASIC.

....

3. In un'istruzione PRINT, com'è chiamata una sequenza di caratteri fra virgolette?

....

4. Qual è lo scopo dell'istruzione REM?

....

5. Se in un programma BASIC c'è un'istruzione READ, quale altro tipo di istruzione deve essere anch'esso presente nel programma?

....

6. Che cosa succede se viene eseguito il seguente programma?

```
100 LET X=3
110 LET Y=4
120 PRINT "Y = ";X
130 END
```

....

7. Quante sono le colonne standard di stampa per riga in BASIC quando le quantità da stampare sono separate da virgole?

....

8. Quante istruzioni DATA vi possono essere in un programma?

....

9. Qual è lo scopo dell'istruzione TAB nel BASIC?

....

10. Come si presenterà l'uscita se verrà eseguito il programma seguente?

```
100 LET A=1
110 LET B=3
120 PRINT A,B
130 PRINT A;B
140 END
```

....

11. Il programma

```
100 INPUT A,B
110 LET C=A+B
120 PRINT C
130 END
```

viene eseguito e in risposta al segnale di input battete 10, 12 e 13. Descrivete esattamente quello che accadrà.

....

12. Le miglia si convertono in chilometri moltiplicando per 1.609. Così, 10 miglia equivalgono a 16.09 chilometri, e così via. Scrivere un programma che produca l'uscita seguente quando è eseguito.

IMMETTERE NUM. MIGLIA (Si immette il numero)
(Il computer stampa il vostro numero) MIGLIA EQUIVALGONO A
(Risposta) CHILOMETRI

....

Capitolo

Decisioni e salti 5

5.1 Obiettivi

La potenza di un computer risiede in larga parte nella sua capacità di prendere decisioni nei programmi. In questo capitolo esploreremo questa caratteristica e proseguiremo a imparare a programmare in BASIC. Gli obiettivi sono i seguenti.

ISTRUZIONI DI SALTO

Le decisioni prese nei programmi possono far sì che il computer salti a numeri di riga al di fuori dell'ordine consecutivo. Tale trasferimento ad una riga di programma può essere incondizionato o può dipendere dai valori di variabili del programma. Queste istruzioni di salto condizionato e incondizionato fanno sì che dei programmi semplici producano risultati potenti e utili.

ESEMPI DI PROGRAMMI

Come nel precedente capitolo, continueremo ad imparare come applicare le tecniche che studiamo in programmi BASIC.

TROVARE ERRORI NEI PROGRAMMI

Quasi tutti i programmi contengono degli errori quando sono scritti la prima volta. Controllare i programmi per trovare degli errori è un'abilità vitale che, come la stessa programmazione, può essere imparata.

5.2 Esercizi di scoperta

1. Accendere il computer e la TV e immettere il programma seguente:

```
100 LET X=1
110 PRINT X
120 LET X=X+1
130 IF X<5 THEN 110
140 END
```

Il simbolo < della riga 130 significa "minore di"; perciò l'istruzione si traduce con "se X è minore di 5 allora 110". Studiate attentamente il programma. Che cosa pensate che sarà visualizzato se si esegue il programma?

....

Eseguite il programma e registrate qui sotto quello che è successo.

....

2. Ora battete

```
100 LET X=2
```

Visualizzate il programma in memoria. Come sarà ora l'uscita?

....

Eseguite il programma e scrivete quello che il computer ha stampato.

....

3. Ora effettuiamo un altro cambiamento nel programma per vedere se state seguendo quello che avviene. Battete

```
120 LET X=X+2
```

Visualizzate il programma e studiatelo attentamente. Che cosa pensate che farà ora?

....

Eseguite il programma e controllate se avevate ragione. Copiate qui sotto quello che è successo in realtà.

....

4. Esploriamo ora un'altra idea in connessione con il programma che avete in memoria, ma abbiamo bisogno di effettuare alcuni cambiamenti. Potete modificare il programma in modo che concordi con quello elencato qui sotto, o potete cancellare il programma in memoria ed immettere quello qui sotto riportato.

```
100 LET X=1
110 PRINT X
120 LET X=X+1
130 IF X>=5 THEN 140
135 GOTO 110
140 END
```

Eseguite il programma e registrate quello che è successo.

....

Confrontate l'uscita registrata qui sopra con quella che avete trascritto nel punto 1. C'è qualche rapporto?

....

5. Visualizzate il programma in memoria. In questo programma c'è un'affermazione enunciata nella riga 130. L'affermazione è X >= 5, che si legge come "X maggiore o uguale a 5". Se per esempio X avesse il valore numerico 6, l'affermazione sarebbe vera. Se X avesse il valore 3, l'affermazione sarebbe falsa.
Ora osserviamo il programma che si trova al punto 4. Se il programma venisse eseguito, il computer inizierebbe con la riga 100, quindi andrebbe alle righe 110, 120 e 130. Se l'affermazione contenuta nella riga 130 è vera, quale numero di riga eseguirà in seguito il computer?

....

6. Finora nei programmi sono state usate soltanto due condizioni. Queste sono

   < (Minore di)
   >= (Maggiore o uguale a)

   Come scrivereste le condizioni:
   Maggiore di

   ....

   Minore o uguale a

   ....

   Uguale a

   ....

   Potete indovinare come si indica "diverso da"?

   ....

   Se sapete completare le risposte qui sopra senza troppe difficoltà, bene. Se no, non preoccupatevi, perché ripasseremo tutto più avanti. La cosa importante ora è capire come lavora l'istruzione IF THEN ("se... allora").

7. Ora passiamo a qualche applicazione che usi le istruzioni IF THEN. Cancellate il programma in memoria e immettete il seguente:

```
100 PRINT "INSERISCI 1,2, O 3 ";
110 INPUT Y
120 IF Y=1 THEN 150
130 IF Y=2 THEN 170
140 IF Y=3 THEN 190
150 PRINT "FRAGOLE"
160 GOTO 100
170 PRINT "LAMPONI"
180 GOTO 100
190 PRINT "PANNA"
200 GOTO 100
210 END
```

   Visualizzare il programma e controllare di averlo immesso in modo esatto. Studiatelo brevemente. Ricordate che, quando il programma

viene eseguito e il computer scrive il segnale di input, dovete battere 1, 2, o 3. Qual è il valore di **Y** che permetterà al computer di raggiungere la riga 150 del programma?

....

Quale valore o valori di **Y** permetteranno al computer di raggiungere la riga 170?

....

E la riga 190?

....

8. Supponiamo che voi vogliate che il computer presenti **LAMPONI**. Quale valore di **Y** dovrà essere immesso?

   ....

   Vedete un po' se avevate ragione. Eseguite il programma e immettete il numero che avete annotato. Che cosa è successo?

   ....

9. Qual è il valore di **Y** che farà sì che il computer visualizzi **FRAGOLE**?

   ....

   E per far stampare **PANNA** dal computer?

   ....

   Controllate ciascuna delle risposte per vedere se avevate ragione.

10. Il programma presume che al segnale di input venga battuto un numero, 1, 2 o 3. Pensate un momento al programma e poi provate ad immaginare che cosa succederebbe se si battesse 4 in risposta al segnale di ingresso. Che cosa pensate che possa succedere?

    ....

    Eseguite il programma, battete **4** in risposta al segnale di ingresso e annotate qui sotto quello che è successo.

    ....

Potete facilmente spiegarvi quello che è successo. Premete RUN/STOP/RESTORE e battete LIST per visualizzare il programma. Considerate quello che il computer fa quando si imbatte nell'affermazione contenuta nell'istruzione IF THEN. Ricordate che se l'affermazione è vera, il computer passa al numero di riga che segue il THEN. Se la condizione è falsa, il computer passa al numero di riga successivo nell'ordine numerico.

11. RUN/STOP O RUN/STOP/RESTORE interrompono il programma ma è possibile scrivere un programma che si possa interrompere con un'unica pressione di tasto. Cancellate il programma in memoria e battete il programma seguente:

```
100 PRINT "PREMETE DEI TASTI"
110 GET A$
120 IF A$= "Q" THEN 150
130 PRINT A$;
140 GOTO 110
150 PRINT "FATTO"
160 END
```

Quando avviate il programma, premete diversi tasti prima di immettere la Q maiuscola. Eseguite il programma. Che cosa è successo?

....

Quante volte viene visualizzato il carattere di ciascun tasto?

....

Premete Q per fermare il programma. La Q viene stampata sullo schermo?

....

12. Cambiate il programma nel modo seguente. Cancellate le righe 130 e 150. Aggiungete la riga 125:

```
125 IF A$="" THEN 110
```

Visualizzate il programma.

....

13. Ora cambiate la riga 120 del programma come segue:

```
120 IF A$<> "" THEN PRINT A$
```

Visualizzate il programma ed eseguitelo. Premete diversi tasti. Cosa succede dopo che avete premuto un tasto?

....

Il computer indica che sta aspettando la pressione di un tasto?

....

Premete la barra di spaziatura diverse volte, poi il tasto K. La barra crea degli spazi?

....

Interrompete il programma con RUN/STOP.

14. Cancellate la riga 160, cioè l'istruzione END. Aggiungete la linea 115:

```
115 IF A$="Q" THEN END
```

Cercate di indovinare quale tasto va premuto per terminare il programma. Eseguitelo e vedete se avete indovinato.

....

Sostituite la Q della riga 115 con uno spazio, come segue:

```
115 IF A$=" " THEN END
```

Quale tasto farà fermare ora il programma?

....

Visualizzate il programma. Ora eseguitelo e premete vari tasti prima di premere la barra spaziatrice. Avevate ragione?

....

15. Cancellate la memoria e immettete ora il seguente programma:

```
100 LET S=0
110 INPUT Y
```

```
120 IF Y=11111 THEN 150
130 LET S=S+Y
140 GOTO 100
150 PRINT S
160 END
```

Questo programma deve sommare i numeri che sono immessi. Il valore di input che provoca la stampa della somma è 11111. Non va a far parte della somma.

16. Avviate con RUN il programma e ogni volta che il segnale di input viene visualizzato, immettete un numero preso dalla seguente sequenza (ricordate di premere RETURN dopo ogni numero):

    3 1 6 5 11111

    Qual è il valore che viene stampato per S?

    ....

    Questo valore di S è la somma dei valori che avete immesso?

    ....

17. Provate e scoprite perché.
    Immettere la riga seguente per studiare il cambiamento del valore di S:

```
135 PRINT " S= ";S
```

Listate il programma, avviatelo e immettete gli stessi valori del punto 16. Che cosa sono i valori di S nella riga 135 in confronto ai valori immessi nella riga 110?

....

Benché possiate già aver scoperto l'errore logico del programma, seguitene attentamente l'esecuzione. Potete vedere che l'istruzione GOTO della riga 140 va alla riga sbagliata, cioè alla riga 100, dove S è rimesso ogni volta a zero.

18. Cancellate la riga 135 e modificate la riga 140 come segue

```
140 GOTO 110
```

Avviate il programma e immettete gli stessi valori (3, 1, 6, 5, 11111). L'uscita è la somma dei numeri che voi avete inserito?

....

Eseguite un'ultima volta il programma con gli stessi valori per verificarne l'esattezza.

19. Spegnete il computer e la TV e passate all'analisi degli obiettivi.

5.3 Analisi

ISTRUZIONI DI SALTO

Fin dall'inizio del libro abbiamo usato istruzioni di salto incondizionato. Il seguente programma ne illustra l'uso:

```
100 LET Z=2
110 PRINT Z
120 LET Z=Z*2
130 GOTO 110
140 END
```

Ricordate che, quando gli si ordina di eseguire un programma BASIC, il computer va all'istruzione che ha il numero di riga più basso e quindi esegue le istruzioni in ordine di riga crescente. L'unico modo di interrompere ciò è per mezzo di un'istruzione di salto (o, come vedremo nel prossimo capitolo, un comando di iterazione). Nel suddetto programma il computer eseguirebbe i numeri di riga nell'ordine seguente: 100, 110, 120, 130, 110, 120, 130, 110, 120, 130, e così via. Il punto importante è che l'istruzione della riga 130 fa sì che il computer salti indietro alla riga 110, invece di andare alla riga 140. Si noti che qui non vi sono condizioni aggiunte all'istruzione della riga 130. Questo perché l'istruzione GOTO è un'istruzione di salto (o di trasferimento) incondizionato. È anche chiaro che l'istruzione GOTO in questo caso mette il programma in un'iterazione e che non c'è modo per uscirne se non interrompendo da tastiera il programma mentre è in esecuzione. (Può darsi che il programma produca un numero troppo grande, che il BASIC non può gestire, provocando un errore di OVERFLOW che lo blocca).
Per riassumere, se ad un certo punto nel programma volete che il computer salti ad un'altra riga senza condizioni, usate l'istruzione GOTO. Però state attenti a non mettere il programma in un circolo chiuso senza possibilità di uscita.

L'istruzione GOTO è incondizionata

Alcune istruzioni di salto contengono condizioni.
A questo punto avrete senza dubbio già fatto il collegamento fra le istruzioni IF THEN che avete incontrato nel lavoro sul computer e la nozione di trasferimento condizionato.

Le istruzioni IF THEN pongono delle condizioni

Tutte le istruzioni di salto condizionato hanno la stessa forma:

N° riga IF *espressione — condizione — espressione* THEN *N° riga*

Notate la forma di una semplice istruzione condizionale:

```
240 IF 3*X - 2 > Y - Z THEN 360
```

Qualunque sia l'affermazione, tutte le istruzioni IF THEN hanno lo stesso formato. L'IF ("se") e il THEN ("allora"), così come i due numeri di riga dell'istruzione non richiedono particolari spiegazioni. Il cuore dell'istruzione si trova nelle due espressioni separate dalla condizione che forma l'affermazione. Dobbiamo osservarle attentamente.
In tutti gli esempi che abbiamo visto finora, con l'eccezione di quello qui sopra, le relazioni sono state delle semplici variabili o delle costanti. Questo è il tipo di confronto usato più spesso nei programmi. Esempi ne possono essere

```
100 IF U<3 THEN 250

340 IF S>T THEN 220
```

Vi sono casi, tuttavia, in cui possiamo aver bisogno di usare delle espressioni più complicate nelle istruzioni IF THEN. Nell'esempio che segue

```
240 IF 3*X - 2 > Y - Z THEN 360
```

la prima relazione è

```
3*X - 2
```

che va bene ammesso che X abbia un valore. La seconda relazione

```
Y - Z
```

può anch'essa essere usata se Y e Z hanno dei valori. Per illustrare meglio quel che accade in un programma, supponiamo che X abbia il valore 1, Y il valore 10 e Z sia 4. Il computer semplificherà l'istruzione sostituendo innanzitutto i valori di X, Y e Z e facendo i calcoli. Questo trasforma l'istruzione in

```
240 IF 1>6 THEN 360
```

Presto o tardi tutte le istruzioni IF THEN si riducono a questa forma, nella quale il computer deve giudicare se un'affermazione stabilita fra due numeri e una condizione è vera o falsa. In questo caso l'affermazione 1>6 è falsa, mentre un'affermazione come 4<10 sarebbe vera. Se l'affermazione è vera, il computer passerà al numero di riga che segue il THEN. Se l'affermazione è falsa, il computer passerà alla riga con il numero immediatamente superiore nel programma.

IF condizione vera THEN salta
IF condizione falsa THEN va al numero di riga successivo

Varie condizioni possono essere usate nell'istruzione IF THEN. Le condizioni con i loro significati sono elencate qui sotto.

Condizione	Significato
=	Uguale a
<	Minore di
>	Maggiore di
<=	Minore o uguale a
>=	Maggiore o uguale a
<>	Diverso da

Le indicazioni che seguono THEN o sono inserite in istruzioni condizionali, non richiedono numeri di linea. Per esempio, sono possibili queste istruzioni:

```
IF X=5 THEN END
IF X>=6 THEN PRINT "HAI PERSO"
```

Più spesso però THEN è seguito dai numeri di linea: la possibilità di saltare in qualsiasi punto del programma è infatti uno strumento molto potente per il programmatore.

5.4 Esempi di programmi

Fino a questo punto i nostri programmi hanno sofferto di gravi limitazioni. Il programma poteva operare delle ripetizioni, ma non c'era poi modo di fermare il procedimento se non interrompendolo. Il programma si fermava, ma spesso aveva la tendenza ad essere banale. Quello che vogliamo è uno strumento per avere un programma che svolga un compito utile (che può coinvolgere la ripetizione) e che poi si fermi. Le istruzioni di salto condizionato che abbiamo appena imparato forniscono un meccanismo per fare questo. Ora vedremo vari programmi; il primo è molto semplice ma gli altri vi daranno un'idea di come far lavorare i programmi.

ESEMPIO 1 — STAMPA DI NUMERI

Il nostro problema è di scrivere un programma che stampi il seguente schema numerico:

```
2   3   4   5
6   7   8   9
10  11
```

Questo schema ha varie caratteristiche da tener presente quando scriviamo il programma. Il primo numero è 2 e i numeri successivi sono spaziati fra loro nella spaziatura standard (quattro numeri per riga). Ciascun numero è più grande del precedente di un'unità. L'ultimo numero stampato è 11, e poi il computer dovrebbe fermarsi.
Varie soluzioni sono possibili: una, non la più elegante, è

```
100 PRINT 2,3,4,5,6,7,8,9,10,11
110 END
```

Potete controllare questo programma e vedere che in realtà produce lo schema di numeri giusto. Illustra anche un concetto molto importante. Non esiste "il programma giusto" in assoluto. L'unico controllo che si può applicare è "Il programma funziona?". Certamente alcuni programmi sono più veloci o sanno raggiungere i risultati in modo più efficiente di altri, ma questo è un altro discorso.
Ora torniamo al problema che stavamo affrontando. Un modo di avvicinarsi al problema è di far sì che il computer stampi il primo numero dello schema. Vogliamo anche organizzare il programma in modo che sia richiesta una sola istruzione di stampa: il programma dovrà stampare il

valore di una variabile che cambierà durante l'esecuzione. Iniziamo il nostro programma con il seguente segmento:

```
100 LET X=2
110 PRINT X,
```

Il valore di X è fissato a 2, e questo valore è stampato nella riga 110. La virgola fa sì che il computer spazi sulla riga fino alla nuova posizione standard di stampa. Ora dobbiamo generare il valore successivo da stampare. (Notate che, in qualunque punto dello schema numerico, il valore successivo è appena 1 più del numero attuale). Questo può essere fatto con

```
120 LET X=X+1
```

Ora tutto quello che resta da fare è far prendere al computer una decisione: tornare indietro all'istruzione di stampa o fermarsi. Finché X è minore di o uguale a 11 vogliamo che torni indietro. Possiamo fare ciò con un'istruzione di salto condizionato.

```
130 IF X<=11 THEN 110
```

Il programma termina con l'istruzione END.

```
140 END
```

Il programma completo è

```
100 LET X=2
110 PRINT X,
120 LET X=X+1
130 IF X<=11 THEN 110
140 END
```

Questo è un programma semplice che non ha alcun valore pratico, se non quello di illustrare come un'istruzione di salto condizionato possa tirarci fuori da un programma al momento opportuno.

ESEMPIO 2 — TASSA DI CIRCOLAZIONE

Facciamo l'ipotesi che voi dobbiate creare un programma per l'ACI, per il calcolo della tassa di circolazione. Le cifre riportate qui sotto sono ovviamente inventate per semplificare le cose.

Potenza in cavalli	Tassa di circolazione
Fino a 50 CV	0
Più di 50 fino a 100 CV	30000
Più di 100 fino a 200 CV	70000
Più di 200 fino a 300 CV	150000
Più di 300 CV	500000

Vogliamo un programma che produca la seguente uscita quando viene eseguito:

POTENZA DELL'AUTO? (Si immette la potenza in CV)
LA TASSA DI CIRCOLAZIONE È (Il computer stampa la tassa)

POTENZA DELL'AUTO? (Si immette la potenza in CV)
LA TASSA DI CIRCOLAZIONE È (Il computer stampa la tassa)

(ecc.)

Chiaramente, l'unica difficoltà del programma è decidere qual è la tassa. Questo procedimento decisionale è fatto attraverso l'istruzione IF THEN. Tanto per cominciare, dobbiamo far sì che si possa immettere la potenza della macchina. Useremo P per indicare questa grandezza. Il programma può dunque cominciare con

```
100 PRINT "POTENZA DELL'AUTO ";
110 INPUT P
```

Ora dobbiamo trovare un metodo per decidere a quale categoria appartiene P. Un modo logico sarebbe di controllare verso l'alto, cominciando dalle potenze più basse. Per prima cosa possiamo controllare se P è 50 o meno. In questo caso sappiamo che la tassa sarà 0.

```
120 IF P<=50 THEN     (La tassa è 0)
```

Il numero di riga che segue THEN manca per il seguente motivo: se il numero contenuto in P è minore di 50 o uguale a 50, vogliamo che il computer salti ad un'istruzione che assegnerà alla tassa il valore 0. Il problema è che a questo punto non sappiamo quale numero di riga dovrà essere usato per questa istruzione. Di conseguenza, lasciamo lo spazio vuoto e ci torneremo su più tardi per inserirvi il valore appropriato. La nota a destra ha la funzione di ricordarci quale dovrebbe essere la tassa se l'affermazione è vera e viene effettuato il salto.

Se l'affermazione della riga 120 è falsa, il computer passa alla riga successiva. In questo caso controlliamo se P ricade nella categoria immediatamente superiore.

```
130 IF P<=100 THEN (La tassa è 30000)
```

Ancora non sappiamo che numero di riga usare dopo THEN, ma potremo completarlo in seguito. Restano ancora tre istruzioni di salto per determinare completamente a quale categoria appartiene P. Esse sono:

```
140 IF P<=200 THEN (La tassa è 70000)
150 IF P<=300 THEN (La tassa è 150000)
160 IF P>300 THEN  (La tassa è 500000)
```

Il programma a questo punto è

```
100 PRINT "POTENZA DELL'AUTO ";
110 INPUT P
120 IF P<=50 THEN  (La tassa è 0)
130 IF P<=100 THEN (La tassa è 30000)
140 IF P<=200 THEN (La tassa è 70000)
150 IF P<=300 THEN (La tassa è 150000)
160 IF P>300 THEN  (La tassa è 500000)
```

Ora possiamo completare immettendo il numero di riga che mancava nella riga 120. Siccome il prossimo numero di riga del programma sarebbe 170, possiamo benissimo usarlo.

```
100 PRINT "POTENZA DELL'AUTO ";
110 INPUT P
120 IF P<=50 THEN 170
130 IF P<=100 THEN (La tassa è 30000)
140 IF P<=200 THEN (La tassa è 70000)
150 IF P<=300 THEN (La tassa è 150000)
160 IF P>300 THEN  (La tassa è 500000)
170 LET F=0
180 GOTO (Istruzione PRINT)
```

Di nuovo, nella riga 180 c'è un numero di riga mancante. Il promemoria dice che vogliamo passare ad un'istruzione di stampa. Se l'affermazione della riga 120 è vera, il computer salta alla riga 170 e assegna il valore 0 a T, che sta per tassa. Possiamo continuare a completare i numeri di riga mancanti nelle righe 130, 140, 150 e 160 usando lo stesso schema. Il risultato è

```
100 PRINT "POTENZA DELL'AUTO ";
110 INPUT P
120 IF P<=50 THEN 170
130 IF P<=100 THEN 190
140 IF P<=200 THEN 210
150 IF P<=300 THEN 230
160 IF P>300 THEN 250
170 LET F=0
180 GOTO       (Istruzione PRINT)
190 LET F=30
200 GOTO       (Istruzione PRINT)
210 LET F=70
220 GOTO       (Istruzione PRINT)
230 LET F=150
240 GOTO       (Istruzione PRINT)
250 LET F=500
```

La riga successiva nel programma sarebbe 260, che possiamo benissimo usare per l'istruzione **PRINT**. Il resto del programma segue facilmente. Il programma completo è dato qui sotto.

```
100 PRINT "POTENZA DELL'AUTO ";
110 INPUT P
120 IF P<=50 THEN 170
130 IF P<=100 THEN 190
140 IF P<=200 THEN 210
150 IF P<=300 THEN 230
160 IF P>300 THEN 250
170 LET F=0
180 GOTO 260
190 LET F=30
200 GOTO 260
210 LET F=70
220 GOTO 260
230 LET F=150
240 GOTO 260
250 LET F=500
260 PRINT "LA TASSA DI CIRCOLAZIONE E' ";F*1000
270 PRINT
280 GOTO 100
290 END
```

Forse avrete notato che l'istruzione di salto condizionato della riga 160 non è necessaria. Per vedere perché, considerate ciascuna delle affermazioni nelle istruzioni **IF THEN**. Se l'affermazione della riga 120 è falsa, sappiamo che **P** dev'essere più grande di 50. Allo stesso modo, se ognuna delle affermazioni successive è falsa, il computer passa alla riga successiva. In particolare, supponiamo che il computer raggiunga la riga 150 e

determini che l'affermazione è falsa. Quindi passa alla riga 160, ma a quel punto sappiamo già che **P** dev'essere maggiore di 300 e che si può perciò stampare la tassa senza nessun'altra prova. Se assegnassimo la tassa di 500000 lire alla riga 160, come risultato si avrebbe un programma leggermente diverso:

```
100 PRINT "POTENZA DELL'AUTO ";
110 INPUT P
120 IF P<=50 THEN 200
130 IF P<=100 THEN 220
140 IF P<=200 THEN 240
150 IF P<=300 THEN 260
160 LET F=500
170 PRINT "LA TASSA DI CIRCOLAZIONE E' ";F*1000
180 PRINT
190 GOTO 100
200 LET F=0
210 GOTO 170
220 LET F=30
230 GOTO 170
240 LET F=70
250 GOTO 170
260 LET F=150
270 GOTO 170
280 END
```

Poiché le operazioni da compiere sono corte, un'altra versione del programma potrebbe essere:

```
100 PRINT "POTENZA DELL'AUTO ";
110 INPUT P
120 IF P<=50 THEN F=0: GOTO 170
130 IF P<=100 THEN F=30: GOTO 170
140 IF P<=200 THEN F=70: GOTO 170
150 IF P<=300 THEN F=150: GOTO 170
160 LET F=500
170 PRINT "LA TASSA DI CIRCOLAZIONE E' ";F*1000
180 PRINT
190 GOTO 100
200 END
```

Notate i due punti nella riga 120: sono usati per separare le istruzioni multiple dopo il THEN in un'istruzione IF THEN.
Tutte e tre le versioni del programma funzionano bene e potreste produrne anche un'altra. Scegliete la versione che preferite; l'unico punto è che il programma funzioni bene.
Studiate il programma fino a che siete sicuri che esegue quello che si de-

siderava fargli eseguire. Cercate anche di ricordare la tecnica di lasciare vuoto lo spazio per il numero di riga quando non sapete quale dovrebbe essere questo numero, tornandoci poi sopra più tardi per completare lo spazio con i valori appropriati. I commenti a destra della riga in questi casi vi aiuteranno a ricordare quello che volete che succeda con il salto a quel punto del programma.

ESEMPIO 3 — MEDIA DI NUMERI

Supponiamo di avere dei numeri in un'istruzione DATA e di volerne ottenere la media. Il problema è che non sappiamo in anticipo quanti numeri vi possano essere. Useremo allora una variabile "semaforo" (flag) per segnare la fine dei dati. Il flag sarà un numero che molto difficilmente potrebbe essere incluso nei dati. Per il nostro flag useremo il numero 9999, ma potreste usarne uno di vostra scelta.
Ecco come funziona. La riga dei dati apparirà sempre così:

N° riga DATA *numero,numero,...,numero,9999*

Ogni volta che un numero viene letto nell'istruzione DATA, il programma deve controllare se questo è 9999. Se non lo è, il numero appena letto fa parte dei dati di cui si vuole ottenere la media. Se il numero è 9999, tutti i dati sono stati letti e si può passare al resto del programma.
La media si calcola dividendo la somma dei numeri per il loro numero. Nel nostro programma dobbiamo calcolare entrambe queste quantità. Useremo S per la somma dei numeri e N per il numero dei numeri. Quando il programma viene eseguito, il computer non sa quali saranno questi valori, e così li pone uguali a zero e quindi sviluppa i loro valori via via che si leggono numeri nelle istruzioni DATA.
Il programma comincia fissando i valori iniziali di S e di N, che in questo caso sono uguali a 0.

```
100 LET S=0
110 LET N=0
```

Ora possiamo leggere un numero dall'istruzione DATA e controllarlo per il valore di flag.

```
120 READ X
130 IF X=9999 THEN       (Calcola la media)
```

Lasciate un numero di riga vuoto nell'istruzione di salto condizionato fi-

no a quando non conoscete quale debba essere. In questo caso, se l'affermazione (X=9999) è vera allora sappiamo che tutti i numeri dell'istruzione DATA sono stati elaborati e che siamo pronti per calcolare la media. Se l'affermazione è falsa, allora il numero appena letto deve far parte dei dati e dev'essere elaborato. Questo si fa nel modo seguente:

```
140 LET S=S+X
150 LET N=N+1
```

Nella riga 140 il valore di X (il numero appena letto) è aggiunto al valore di S. Ricordate che la somma di tutti i numeri di cui si deve ottenere la media si sta sviluppando in S. Nella riga 150 il numero N viene incrementato di 1 per registrare il fatto che un altro numero è stato elaborato. Ora aggiungete un GOTO all'istruzione READ, che ha il compito di far cercare al computer il successivo numero nei DATA.

```
160 GOTO 120
```

Ora possiamo completare la riga 130 inserendovi il numero di riga che mancava, giacché il prossimo numero di riga sarebbe normalmente 170. Nella riga 170 calcoliamo la media, che identificheremo con M. Se viene aggiunta una istruzione DATA tipica, il programma completo è

```
100 LET S=0
110 LET N=0
120 READ X
130 IF X=9999 THEN 170
140 LET S=S+X
150 LET N=N+1
160 GOTO 120
170 LET A=S/N
180 PRINT A
190 DATA 4,2,3,6,5,9999
200 END
```

Naturalmente possiamo avere tutte le istruzioni DATA che sono necessarie per contenere i numeri di cui si deve fare la media. Dopo l'ultimo numero nell'ultima istruzione DATA mettiamo il flag 9999 per segnare la fine dei dati. Questo ci fa uscire dall'iterazione di lettura (READ) e ci permette di sapere quando passare al calcolo della media. L'istruzione di salto condizionato unita all'idea di una variabile che faccia da "semaforo" ci fornisce un potente strumento da usare nei programmi.

TROVARE GLI ERRORI NEI PROGRAMMI

La capacità di osservare un programma e di determinare se fa quello che dovrebbe fare è certo una delle più importanti che il principiante possa acquisire. Per venire più al sodo, dovete riuscire a trovare quello che non va e a correggerlo quando un programma non fa quello che dovrebbe fare.

Controllare i programmi e trovarne gli errori significa svolgere due compiti separati. Primo, dovete decidere quali sono le variabili di cui vorreste avere maggiori informazioni. Potete inserire delle istruzioni PRINT nel programma per stampare i valori delle variabili a cui siete interessati. Secondo, dovete essere in grado di seguire la logica del programma, seguendolo passo passo (usando carta e penna, se necessario). Nei loop è generalmente sufficiente controllare la prima e l'ultima coppia di valori. Queste due operazioni portano spesso ad una rapida determinazione degli errori logici di programmazione.

5.5 Problemi

1. Scrivere un programma BASIC che richieda l'immissione di due numeri e quindi stampi il maggiore dei due.

2. Scrivere un programma BASIC che legga (READ) tre numeri da un'istruzione DATA e poi stampi il più piccolo dei tre.

3. Scrivere un programma per calcolare e stampare la somma di tutti i numeri interi compresi fra 1 e 100, estremi inclusi.

4. Che cosa succederebbe se il seguente programma venisse eseguito? Descrivetelo con parole vostre.

```
100 LET S=0
110 LET X=1
120 LET S=S+X
130 LET X=X+2
140 IF X<100 THEN 120
150 PRINT S
160 END
```

5. Nell'esempio 3 del paragrafo 5.4, modificate nel modo seguente la riga 190:

```
190 DATA 4,2,3,6,5,1111
```

Studiate il programma e annotate per iscritto quello che verrebbe visualizzato se il programma venisse eseguito. Sarà bene che proviate ad eseguire il programma per vedere se avevate ragione. Può essere utile inserire la riga 155 come segue:

```
155 PRINT "S = ";S;" N = ";N
```

6. Scrivere un programma per trovare la media di tutti i numeri positivi di un elenco la cui fine è segnata dal flag 999. La lista dei numeri dev'essere immessa nel computer durante l'esecuzione.

7. Supponiamo che vi sia un'istruzione DATA che contiene una lista di numeri di lunghezza ignota. La fine della lista però è segnata con il flag 9999. Scrivere un programma BASIC per calcolare e stampare la somma dei numeri della lista il cui valore sia compreso fra -10 e $+10$ estremi compresi.

8. Di solito nei supermercati si aggiunge alle merci un ricarico che varia a seconda del costo unitario del prodotto. Si supponga che questo ricarico sia basato sulla tabella seguente:

Costo unitario	**Ricarico**
da 1 a 1000 lire	20%
da 1001 a 2000 lire	10%
oltre 2000 lire	5%

Il costo unitario è determinato dividendo il prezzo della confezione per il numero dei pezzi contenuti. Scrivere un programma che calcoli il prezzo da segnare sul cartellino, cioè il costo unitario più il ricarico. Stabilite i messaggi per l'immissione e gli ingressi come volete.

9. Supponiamo che facciate un accordo per lavorare per 10 mila lire il primo giorno, 20 mila lire il secondo giorno, 40 mila lire il terzo, 80 mila lire il quarto e così via. Siccome in un mese ci sono 22 giornate lavorative, scrivete un programma che calcoli la vostra paga per un mese di lavoro.

10. Considerate la serie

$1+1/2+1/3+1/4+...$

Scrivere un programma per trovare la somma dei primi N termini. Usatelo per trovare la somma dei primi 10, 100 e 1000 termini. Basandovi su questi risultati, quale pensate che sarà il risultato se lasciassimo andare avanti la serie all'infinito?

11. Studiate il seguente programma. Sapete descrivere quello che fa? Sarà bene che usiate le tecniche di scoperta degli errori per risolvere il problema.

```
100 READ N
110 LET L=1
120 LET C=1
130 READ X
140 LET C=C + 1
150 IF X<L THEN 170
160 LET L=X
170 IF C<N THEN 130
180 PRINT L
190 DATA 10
200 DATA 5,83,17,3,47
210 DATA 25,16,41,51,7
220 END
```

Ulteriori informazioni su come il programma funziona possono essere ottenute inserendo la riga 165 come segue

```
165 PRINT "L E' ";L
```

e avviando il programma.

12. Il programma seguente dovrebbe trovare la media di N numeri battuti al terminale. Così com'è, il programma è sbagliato. Che cosa c'è che non va?

```
100 PRINT "QUANTI NUMERI"
110 INPUT N
120 LET S=0
130 LET C=1
140 PRINT "INSERISCI UN NUMERO";
150 INPUT X
160 LET S=S + X
170 LET C=C + 1
180 IF C<N THEN 140
190 LET A=S/N
200 PRINT "LA MEDIA E' ";A
210 END
```

13. Il prezzo scontato di un prodotto può essere calcolato con

    D = L*(1 − R/100)

    dove L è il prezzo d'acquisto e R è il tasso di sconto in percentuale. Scrivere un programma che produca la seguente stampa quando viene eseguito:

    PREZZO DI LISTINO (L) ? (Si immette il prezzo)
    TASSO DI SCONTO (%) ? (Si immette il tasso)
    IL PREZZO SCONTATO È
    (Il computer emette il prezzo) LIRE

14. C'è una sequenza di numeri piuttosto interessante, chiamata "serie di Fibonacci". Il gruppo comincia con 0, 1. Poi ogni numero successivo della sequenza è la somma dei due precedenti:

    0, 1, 1, 2, 3, 5, 8, ...

    Scrivere un programma BASIC che calcoli e stampi i primi venti numeri della sequenza di Fibonacci.

15. Scrivere un programma che accetti l'immissione di due numeri. Se entrambi i numeri sono maggiori o uguali a 10, stampare la somma. Se entrambi i numeri sono minori di 10, stampare il prodotto. Se un numero è maggiore o uguale a 10 e l'altro è minore, stampare la differenza fra il maggiore e il minore.

16. Un professore decide di usare delle lettere per dare i voti ad un esame, nel modo seguente:

30	A
26-29	B
23-25	C
18-22	D
0-17	F

    Scrivere un programma che, quando avviato, dia il seguente prodotto:

    IMMETTERE IL VOTO D'ESAME? (Si immette il voto numerico)
    IL VOTO È (Il computer emette A, B, C, D, E o F)

17. Se ogni anno usate l'8 per cento in più di elettricità, in nove anni il

vostro consumo raddoppierà. C'è una regola molto interessante, chiamata "regola del settantadue" che può essere usata per calcolare i tempi di raddoppio. Se una quantità cresce di R per cento in un certo periodo di tempo, allora il numero di periodi per cui la quantità raddoppia è dato approssimativamente da 72/R. Possiamo calcolare la crescita direttamente sul computer. In un unico periodo una quantità Q cresce in accordo con la relazione

$$Q_{nuovo} = Q_{vecchio}(1 + R/100)$$

Possiamo seguire la crescita usando ripetutamente la suddetta relazione. Il tempo di raddoppio è il numero di periodi necessari al raddoppio del valore di Q. Applicando questa definizione, scrivete un programma che produca la seguente uscita:

TASSO DI CRESCITA (%)? (Si immette R)
IL NUMERO DEI PERIODI DI CRESCITA
PER IL RADDOPPIO È (Il computer scrive la risposta)

Usate il programma per controllare la precisione della regola del settantadue per tassi diversi di crescita.

18. Un gruppo di numeri interi è scelto a caso dal gruppo 1, 2, 3, 4 e messo in un'istruzione DATA. La fine del gruppo è segnata con il flag 9999. Scrivere un programma BASIC che calcoli e stampi il numero di volte in cui compare l'uno, il due, il tre e il quattro nel gruppo. Provate il vostro programma sulla seguente istruzione DATA:

```
DATA 3,1,2,1,4,4,1,2,2,2,3,9999
```

5.6 Test di apprendimento

1. Quale sarà il risultato se verrà eseguito il seguente programma?

```
100 LET Y=3
110 LET X=2*Y
120 PRINT X
130 LET Y=Y+2
140 IF Y<=10 THEN 110
150 END
```

....

2. Che cosa sarà emesso in uscita se verrà eseguito il seguente programma?

```
100 READ X
110 DATA 1,2,3
120 IF X<2 THEN 160
130 IF X=2 THEN 150
140 PRINT "BUONO"
150 PRINT "MIGLIORE"
160 PRINT "OTTIMO"
170 PRINT
180 GOTO 100
190 END
```

....

3. Supponete di decidere di comprare un certo numero di bottiglie di vino. Il produttore spinge le vendite offrendo dei prezzi scontati per acquisti in quantità. Il dettaglio dei prezzi è il seguente:

Num. pezzi comprati	**Prezzo per bottiglia**
20 o meno	2000 lire
da 21 a 50	1800 lire
51 o più	1500 lire

Scrivere un programma che produca la seguente uscita quando viene eseguito e poi continui a chiedere un nuovo input.

QUANTE BOTTIGLIE? (Si immette la quantità)
IL PREZZO PER BOTTIGLIA È (Il computer scrive il prezzo unitario)
IL COSTO TOTALE DELL'ORDINE È (Il computer scrive il prezzo totale)

....

4. Scrivere un programma che stampi lo schema numerico mostrato qui sotto e quindi si fermi.

```
0  5  10 15 20 25
30 35 40 45 50 55
```

(ecc.)

```
150 155 160 165 170 175
```

Capitolo

Iterazioni e funzioni

6

6.1 Obiettivi

ITERAZIONI

Abbiamo già imparato come fare eseguire delle iterazioni (loop) nei programmi usando sia le istruzioni di trasferimento incondizionato, che quelle di salto condizionato. Il BASIC del C-64 ha delle istruzioni speciali per fornire automaticamente l'iterazione. Queste istruzioni semplificano il compito del programmatore e aggiungono flessibilità ai programmi.

FUNZIONI INCORPORATE

Il BASIC contiene un certo numero di funzioni incorporate che possono essere richiamate per eseguire dei compiti specifici. Studieremo alcune delle funzioni più semplici che riguardano i calcoli numerici e vedremo come esse possano essere usate utilmente per i nostri programmi.

ESEMPI DI PROGRAMMI

Continueremo le attività per accrescere la pratica di programmazione. Ricordate che l'obiettivo generale del libro è quello di insegnarvi come si scrivono i programmi in BASIC.

6.2 Esercizi di scoperta

1. Accendete il computer e la TV. Immettete il seguente programma:

```
100 LET Y=10
110 PRINT Y;
120 LET Y=Y+5
130 IF Y<=50 THEN 110
140 END
```

Studiate il programma ed eseguitelo. Annotate quello che è successo.

....

Quale istruzione nel programma determina la differenza nei numeri che sono stati stampati?

....

2. Cancellate il programma in memoria. Ora introducete il programma seguente:

```
100 FOR Y=10 TO 50 STEP 5
110 PRINT Y;
120 NEXT Y
130 END
```

Eseguite il programma e segnate qui sotto quello che è successo.

....

Confrontate l'uscita con quella ottenuta dal programma del punto 1.

3. Dal momento che i due programmi appena eseguiti producono la stessa uscita, è ragionevole presumere che le istruzioni devono in qualche modo essere in relazione. Battete

```
100 FOR Y=10 TO 50 STEP 10
```

Visualizzate il programma in memoria e studiatelo. Sapendo che FOR significa "per", TO vuol dire "a", STEP "passo" e NEXT

"successivo" che cosa pensate che succederà quando si esegue il programma?

....

Controllate se avevate ragione. Eseguite il programma e segnate qui sotto i risultati.

....

4. Ora battete

```
100 FOR Y=0 TO 5 STEP 1
```

Visualizzate il programma. Che cosa pensate che faccia?

....

Eseguite il programma e scrivetene l'uscita.

....

5. Ora battete

```
100 FOR Y=0 TO 5
```

Visualizzate il programma. Che cosa pensate che faccia?

....

Avviate il programma e annotate quello che è successo.

....

Ora confrontate la riga 100 del programma appena eseguito con la riga 100 del programma del punto 4. Se la differenza fra i numeri che devono essere stampati è 1, è necessaria la parte STEP dell'istruzione?

....

6. Proviamo una tattica diversa. Battete

```
100 FOR Y=20 TO 10 STEP -2
```

Visualizzate il programma e studiatelo. Che cosa pensate che faccia questo programma?

....

Eseguite il programma e registratene qui sotto l'uscita.

....

7. Ora battete

```
100 FOR Y=10 TO 20 STEP -2
```

Visualizzate il programma. Che cosa pensate che succeda ora se si esegue il programma?

....

Eseguite il programma e segnate qui sotto quanto è avvenuto.

....

Notate che il loop FOR NEXT è eseguito una volta, dato che il 10 viene stampato. Siete caduti in una trappola BASIC. Quale può essere il problema?

....

8. Finora le dimensioni dei passi (STEP) nelle istruzioni FOR NEXT hanno funzionato senza problemi. Ora battete

```
100 FOR Y=2 TO 9 STEP 3
```

Visualizzate il programma. Annotate quello che pensate che verrà stampato.

....

Eseguite il programma e segnatevi quello che è successo.

....

9. Passeremo ora ad alcune situazioni più complesse con le istruzioni

FOR NEXT. Cancellate il programma in memoria e immettete il programma seguente:

```
100 FOR X=1 TO 3
110 FOR Y=1 TO 4
120 PRINT X,Y
130 NEXT Y
140 NEXT X
150 END
```

Eseguite il programma e registratene qui sotto l'uscita.

....

10. Ora battete

```
100 FOR X=1 TO 2
```

Eseguite questo nuovo programma e registratene l'uscita.

....

Confrontate i due schemi numerici che avete appena ottenuto. Riuscite a scorgere il rapporto fra lo schema e i limiti delle istruzioni FOR NEXT?

....

11. Modifichiamo ancora un po' il programma. Battete

```
100 FOR X=1 TO 3
110 FOR Y=1 TO 2
```

Visualizzate il programma e studiatelo. Come pensate che si presenterà l'uscita quando verrà eseguito?

....

Provatelo e controllate se avevate ragione.

....

12. Ancora una volta. Battete

```
100 FOR X=1 TO 2
110 FOR Y=1 TO 2
```

Visualizzate il programma e scrivete quello che pensate che verrà stampato quanto il programma verrà eseguito.

....

Eseguite il programma e registratene qui sotto il risultato.

....

Visualizzate il programma appena eseguito. Immaginate di tracciare una linea che vada dal numero di riga dell'istruzione FOR Y al numero di riga dell'istruzione NEXT Y. Fate la stessa cosa per le istruzioni FOR X e NEXT X. Queste linee si incrociano?

....

13. Ora battete

```
100 FOR Y=1 TO 2
110 FOR X=1 TO 2
```

Visualizzate il programma. Quale pensate che sia l'uscita di questo programma?

....

Eseguite il programma e scrivete che cosa è successo.

....

Sul listato delle istruzioni del programma collegate il numero di riga di FOR Y con quello di NEXT Y, come avete fatto al punto 12. Fate la stessa cosa per le istruzioni FOR X e NEXT X. Le linee si incrociano? Confrontate con la stessa situazione al punto 12.

....

La scritta NEXT WITHOUT FOR ERROR vi suggerisce un modo per evitare di mettervi nei pasticci nell'uso di più di una combinazione di FOR NEXT in un unico programma?

....

14. Nel capitolo precedente abbiamo fatto esperimenti con la funzione TAB per ottenere spaziature variabili in uscita. Ora che abbiamo le istruzioni FOR NEXT a nostra disposizione, torniamo alla funzione TAB. Cancellate il programma in memoria e immettete il seguente:

```
100 FOR X=1 TO 5
110 PRINT TAB(X);
120 FOR Y=X TO 5
130 PRINT "Y";
140 NEXT Y
150 PRINT
160 NEXT X
170 END
```

Studiate con calma il programma. Che cosa pensate che produrrà in uscita?

....

Controllate se avevate ragione. Eseguite il programma e annotatene qui sotto l'uscita.

....

15. Cancellate il programma dalla memoria. Immettete il programma che segue.

```
100 INPUT A
110 LET B=SQR(A)
120 PRINT B
130 GOTO 100
140 END
```

Eseguite il programma e al segnale di input, battete 4. Che cosa è successo?

....

Ora immettete 9 e segnate il risultato.

....

Ancora una volta. Battete 25. Che cosa è successo?

....

Per finire, immettete 10. Che cosa è successo?

....

Bene, che succede ad A nell'espressione SQR(A) della riga 110 del programma? In altre parole, che cosa fa SQR?

....

16. Fate uscire il computer dall'iterazione di input. Battete

```
110 LET B=INT(A)
```

Eseguite il programma per i seguenti valori di A. In ciascun caso annotate la risposta del programma.

A	Uscita
1	______
3.4	______
256.78	______
0	______
−1	______
−2.3	______

Esaminate l'uscita che avete annotato qui sopra e confrontate ciascun numero con il corrispondente valore di A che avete immesso. Che cosa fa la funzione INT(A)?

....

Se avete avuto difficoltà nel capire che cosa succedeva ai valori negativi di A, non ve ne preoccupate a questo punto. Faremo un ripasso completo dell'argomento più tardi.

17. Fate uscire il computer dall'iterazione di ingresso. Battete

```
110 LET B=SGN(A)
```

Visualizzate il programma e ripassatene la struttura per rinfrescarvi la memoria su come funziona. Eseguite il programma per ognuno dei valori di A. Per ciascun caso registratene l'uscita.

A	Uscita
1.5	______
43	______
128.3	______
0	______
−1	______
−1.2	______
−345.7	______
4.7	______
−5.8	______

Esaminate le uscite suddette attentamente. Che cosa fa la funzione SGN?

....

18. Un'ultima funzione. Fate uscire il computer dall'iterazione di ingresso. Battete

```
110 LET B=ABS(A)
```

Eseguite il programma per ciascuno dei valori di A dati qui sotto. Di nuovo registrate per iscritto l'uscita in ciascun caso.

A	Uscita
3.4	______
0	______
−3.4	______
−2	______
−8.45	______
8.45	______

Esaminate l'uscita. Che cosa fa la funzione ABS?

....

Interrompete il programma.

19. E con ciò si conclude il lavoro sul computer per ora. Spegnete il computer e la TV e passate alla sezione successiva.

6.3 Analisi

ITERAZIONI

Nei capitoli precedenti abbiamo imparato come fare iterare i programmi sotto il controllo di istruzioni di salto. L'istruzione incondizionata (GOTO) è utile, ma può a volte dar luogo ad un'iterazione senza via d'uscita. Le istruzioni condizionate (IF THEN) forniscono un modo per fare iterare il programma come si desidera e anche un modo per uscirne. Tutte queste tecniche sono buone. Il BASIC però ha un modo molto elegante di occuparsi delle iterazioni togliendo un grosso peso dalle spalle del programmatore. Analizzeremo ora questo nuovo metodo che usa le istruzioni FOR NEXT.
Tutte le istruzioni FOR hanno la stessa struttura. Questa struttura e un'istruzione tipica sono illustrati qui sotto.

N° riga FOR *variabile=espressione* TO *espressione* STEP *espressione*

```
120 FOR X=1 TO 9 STEP 2
```

Nelle istruzioni FOR possono cambiare la variabile e le tre espressioni. Se lo STEP viene lasciato fuori dall'istruzione, il computer userà un passo =1. Possiamo scrivere molte forme diverse di istruzione FOR. Qui sotto ne sono riportate alcune per illustrare la gamma delle possibilità.

```
130 FOR J=2 TO 8
130 FOR T=25 TO 10 STEP -2
130 FOR W=-20 TO 10 STEP 2
130 FOR X=3*Z TO A*B STEP D
```

In generale possiamo scrivere qualunque istruzione BASIC legale nei rapporti coinvolti nell'istruzione FOR, ammesso che le variabili usate siano state naturalmente definite in modo appropriato nel programma.
L'istruzione FOR apre un'iterazione. L'istruzione NEXT chiude l'iterazione.

```
200 FOR X=2 TO 18 STEP 2  (Apre l'iterazione)
```

Righe di programma all'interno dell'iterazione.

.
.
.

340 NEXT X (Chiude l'iterazione)

Nell'istruzione NEXT la variabile dev'essere la stessa dell'istruzione FOR che ha aperto l'iterazione.

FOR apre l'iterazione

NEXT chiude l'iterazione

È importante capire a fondo come queste iterazioni funzionano. Nell'esempio suddetto, quando il programma raggiunge la riga 200 la prima volta, X è posto uguale a 2. Poi il computer esegue le altre righe fino a che non raggiunge la riga 340. Questa chiude l'iterazione e dirige il computer di nuovo alla riga 200 e al successivo valore di X, in questo caso 4. Il computer sta nell'iterazione fino a che il valore di X non eccede il limite di 18. Poi, invece di passare attraverso le istruzioni all'interno dell'iterazione, il computer salta al numero di riga che segue l'istruzione NEXT usata per chiudere l'iterazione. Osserviamo un esempio per vedere ancora una volta le istruzioni FOR NEXT in azione.

```
100 LET A=1
110 FOR X=1 TO 6 STEP 2
120 LET A=2*A
130 PRINT A,X
140 NEXT X
150 END
```

Siccome in questo programma sono coinvolte soltanto due variabili (A e X), elencheremo i numeri di riga nell'ordine in cui il computer le incontra e i corrispondenti valori delle variabili.

Numero di riga	A	X
100	1	
110	1	1
120	2	1
130	2	1
140	2	1
110	2	3
120	4	3
130	4	3
140	4	3
110	4	3
120	8	5
130	8	5
140	8	5
110	8	7*
150**		

*Esce dall'iterazione
**Il programma si ferma

Studiate la sequenza di numeri di riga e i corrispondenti valori di A e X fino a che siete certi di capire come le istruzioni FOR NEXT controllano l'iterazione.
Quando il programma gira, la linea 130 stamperà:

```
2  1
4  3
8  5
```

Sovente in un programma sono richieste delle strutture di iterazione più complicate. La struttura può essere complicata quanto si vuole ammesso però che le iterazioni non si incrocino. L'esempio seguente illustra un segmento di programma con delle iterazioni incrociate.

Un altro esempio di iterazioni incrociate è

L'esempio seguente illustra una complicata struttura di iterazioni organizzate in modo corretto:

In questo esempio abbiamo delle iterazioni doppie e iterazioni all'interno di iterazioni. Ricordate, tuttavia, che qualunque combinazione può essere usata in un programma purché le linee che collegano le istruzioni FOR e

le loro corrispodenti NEXT non si incrocino. Se si incrociano, il computer segnala un errore e si ferma.

Non incrociate le iterazioni FOR NEXT!

FUNZIONI INCORPORATE

Uno dei vantaggi di un moderno computer è che gruppi di istruzioni possono essere programmati in anticipo per svolgere determinati compiti. Numerose funzioni BASIC incorporate permettono al programmatore di eseguire delle operazioni matematiche molto complicate senza difficoltà. La tabella qui sotto mostra diverse di queste funzioni.

Funzione	Azione
SQR(X)	Radice quadrata di X
INT(X)	Parte intera di X
SGN(X)	Segno di X
ABS(X)	Valore assoluto di X

La prima funzione, SQR(X), illustra come funzionano in generale le funzioni. Per prima cosa, X viene chiamato "argomento" della funzione, cioè "ciò su cui la funzione opera". Se in un programma usiamo SQR(X), diamo istruzione al computer di prendere il valore di X ed estrarre la radice quadrata del numero. Per esempio

SQR(36) = 6
SQR(64) = 8
SQR(81) = 9
SQR(2) = +1.41421356

e così via. L'unica limitazione è che non possiamo estrarre la radice quadrata di un numero negativo. Se il computer cercasse di calcolare SQR(−6) per esempio, segnalerebbe ILLEGAL QUANTITY ERROR (quantità non ammessa) e si fermerebbe.
L'argomento della funzione può essere tanto complicato quanto è necessario che lo sia nel programma. Se il computer si imbatte in un'espressione del tipo

SQR(X+4*Y)

cercherebbe i valori di X e di Y, eseguirebbe i calcoli indicati e quindi

estrarrebbe la radice quadrata. Questa caratteristica è valida per tutte le funzioni.
INT(X) prende la parte intera di X. Il termine "intero" è un modo per dire "numero intero". Così, 2 è un intero, mentre 23.472 non lo è. Se prendiamo la parte intera di un numero, ci dimentichiamo semplicemente di tutto quello che segue il punto decimale. Così

INT(3.1593) = 3
INT(54.76) = 54
INT(0.362) = 0

I numeri negativi però richiedono speciale attenzione. Quello che succede veramente quando si prende la parte intera di un numero è che si va al primo numero intero che sia uguale a quel numero o inferiore ad esso. Secondo questa regola vediamo che

INT(−2) = −2
INT(−0.93) = −1

e così via. Si noti attentamente che la funzione INT non arrotonda un numero. Spesso i principianti hanno le idee poco chiare in proposito.
SGN(X) è una funzione molto interessante. Se X (l'argomento della funzione) è positivo, SGN(X), è +1. Se X è negativo, SGN(X) è −1. Se X è 0, SGN(X) è 0. In effetti, SGN(X) fornisce il segno di X, sotto forma di +1, −1, o 0. Perciò

SGN(4.568) = +1
SGN(375) = +1
SGN(0) = 0
SGN(−5.93) = −1
SGN(−4) = −1

A questo punto vi può non essere chiaro perché una tale funzione possa essere utile. In realtà la funzione SGN è molto utile e ha varie applicazioni. Per ora ci accontenteremo di imparare come la funzione lavora.
ABS(X) dice semplicemente al computer di ignorare il segno di X. In effetti, converte tutti i valori di X, ad eccezione dello 0, in numeri positivi. Così

ABS(4.5) = 4.5
ABS(−4.5) = 4.5
ABS(95.34) = 95.34
ABS(−95.34) = 95.34
ABS(0) = 0

Ci sono molte altre funzioni incorporate nel BASIC. La maggior parte di queste, tuttavia, implica conoscenze matematiche superiori a quelle presunte per questo libro. Se avete le conoscenze matematiche che vi permettano di capire che cosa fanno le funzioni, non avrete difficoltà ad imparare ad usarle: fate riferimento al manuale di consultazione del C-64. Nel prossimo capitolo considereremo alcune funzioni che agiscono con le stringhe di caratteri.
Le funzioni incorporate che abbiamo trattato sono usate nelle istruzioni BASIC. Esempi di righe di programma che utilizzano tali funzioni potrebbero essere

```
100 LET X=SQR(Y)
100 LET Z=3*INT(C)+ABS(D)
```

Le funzioni incorporate possono essere usate all'interno di funzioni, come

```
100 LET Y=INT(SQR(X)+3*ABS(Z))
```

Qualunque espressione BASIC può essere argomento di funzioni BASIC

6.4 Esempi di programmi

I programmi seguenti mostrano come usare l'iterazione automatica e le funzioni incorporate per rendere più facile la programmazione.

ESEMPIO 1 — MEDIA DI NUMERI

Nel capitolo precedente, abbiamo usato come esempio di programma il problema di trovare la media. Torniamo allo stesso problema ma usiamo questa volta un metodo diverso. Vogliamo che il programma, quando viene eseguito, produca la seguente uscita:

QUANTI NUMERI? (Si immette)
INSERIRE I NUMERI, UNO PER VOLTA
? (Si immettono i numeri)
LA MEDIA È (Il computer scrive la media)

Le prime righe dovrebbero ora essere facili da scrivere.

```
100 PRINT "QUANTI NUMERI ";
```

```
110 INPUT N
120 PRINT "INSERIRE I NUMERI,UNO PER VOLTA"
```

Dobbiamo ora provvedere all'immissione di N numeri, ma dobbiamo anche tenere a mente che dovremo calcolare la media dei numeri. Così, all'inizio renderemo S (che sarà usata per sommare i numeri) uguale a 0.

```
130 LET S=0
```

L'immissione di N numeri e la somma di essi è un compito ideale per le istruzioni FOR NEXT.

```
140 FOR I=1 TO N
150 INPUT X
160 LET S=S+X
170 NEXT I
```

Notate che non usiamo I, la variabile dell'iterazione, altro che per contare i numeri man mano che vengono immessi. Quando tutti i numeri sono stati immessi, il computer uscirà dall'iterazione per passare al numero di riga successivo al 170. Quando ciò succede, S contiene la somma di tutti i valori di X che sono stati immessi. Siccome sappiamo che N è il numero di numeri che sono stati immessi, siamo immediatamente in grado di calcolare la media.

```
180 LET A=S/N
```

Il resto del programma segue senza difficoltà.

```
190 PRINT "LA MEDIA E'";A
200 END
```

Il programma completo è

```
100 PRINT "QUANTI NUMERI ";
110 INPUT N
120 PRINT "INSERIRE I NUMERI,UNO PER VOLTA"
130 LET S=0
140 FOR I=1 TO N
150 INPUT X
160 LET S=S+X
170 NEXT I
180 LET A=S/N
190 PRINT "LA MEDIA E'";A
200 END
```

ESEMPIO 2 — TABELLA DI CONVERSIONE DELLE TEMPERATURE

In uno dei precedenti programmi abbiamo usato la relazione

$$C = 5/9*(F-32)$$

per convertire i gradi Fahrenheit in gradi Celsius. Ora genereremo una tabella di conversione come segue:

Gradi F	Gradi C
0	−17.77777
5	−15
10	−12.22222
(ecc.)	
100	37.77777

Per prima cosa dovremo stampare l'intestazione e la riga vuota; dopo, la tabella vera e propria.

```
100 PRINT "GRADI F","GRADI C"
110 PRINT
```

Possiamo usare un'iterazione FOR NEXT per generare i valori di F che possono essere convertiti in C e stampati.

```
120 FOR F=0 TO 100 STEP 10
130 LET C=5*(F-32)/9
140 PRINT F,C
150 NEXT F
```

Infine abbiamo bisogno dell'istruzione END.

```
160 END
```

L'intero programma è dato qui sotto.

```
100 PRINT "GRADI F","GRADI C"
110 PRINT
120 FOR F=0 TO 100 STEP 10
130 LET C=5*(F-32)/9
140 PRINT F,C
150 NEXT F
160 END
```

ESEMPIO 3 — DIVISIONE ESATTA

Vogliamo ora scrivere un programma che calcoli tutti gli interi che dividono esattamente un altro numero intero. Per illustrare meglio la cosa, supponiamo di prendere 8 come numero intero di prova. Il problema è quello di trovare tutti i numeri interi che divideranno esattamente 8 senza nessun resto. La regola è

Se N/X = INT(N/X) allora non c'è resto
Se N/X < > INT(N/X) allora c'è un resto

Scriviamo ora un programma che produca l'uscita seguente quando viene eseguito:

INSERIRE UN NUMERO INTERO POSITIVO? (Si immette)
I NUMERI CHE DIVIDONO (il numero) ESATTAMENTE SONO
(Il computer scrive il primo numero,
il secondo numero, ecc.)

Il programma comincia in modo facile.

```
100 PRINT "INSERIRE UN NUMERO INTERO POSITIVO ";
110 INPUT N
120 PRINT "I NUMERI CHE DIVIDONO";N;"ESATTAMENTE SONO"
```

Ora dobbiamo cercare tutti i numeri interi fra 1 e N. Naturalmente questo è l'ideale per le iterazioni FOR NEXT. Usando la regola data sopra, possiamo vedere se ciascun numero divide in modo esatto il numero immesso.

```
130 FOR X=1 TO N
140 IF N/X <> INT(N/X) THEN 160
150 PRINT X,
160 NEXT X
```

Infine abbiamo bisogno di un'istruzione END.

```
170 END
```

Il programma completo è

```
100 PRINT "INSERIRE UN NUMERO INTERO POSITIVO ";
110 INPUT N
```

```
120 PRINT "I NUMERI CHE DIVIDONO";N;"ESATTAMENTE SONO"
130 FOR X=1 TO N
140 IF N/X <> INT(N/X) THEN 160
150 PRINT X,
160 NEXT X
170 END
```

Potreste provare il programma usando dei valori piuttosto grandi di N. Come fareste per far eseguire il programma in metà tempo?

ESEMPIO 4 — PIANO DI AMMORTAMENTO

Quando una ditta fa degli investimenti in attrezzature, l'investimento viene ammortizzato in un certo numero di anni a scopi fiscali. Questo significa che il valore delle attrezzature decresce ogni anno (a causa dell'uso, del consumo) e l'ammontare della diminuzione è un elemento deducibile dalle tasse. Uno dei metodi usati per calcolare l'ammortamento è lo schema della "somma delle cifre degli anni". Per illustrare meglio il concetto, supponiamo che un macchinario abbia una vita media di 5 anni. La somma delle cifre degli anni sarebbe

$$1+2+3+4+5 = 15$$

L'ammortamento sarà il primo anno di 5/15 del valore iniziale. La quota di ammortamento del secondo anno sarà 4/15, e così via. Se il macchinario ha un valore iniziale di 3 milioni di lire, il piano di ammortamento è il seguente:

Anno	Quota di ammortamento	Ammortamento	Valore corrente del bene
1	5/15	1000000	2000000
2	4/15	800000	1200000
3	3/15	600000	600000
4	2/15	400000	200000
5	1/15	200000	0

Il nostro problema è di scrivere un programma BASIC per generare dei piani di ammortamento con il metodo della somma delle cifre degli anni. L'uscita dovrebbe essere così:

QUAL È IL VALORE INIZIALE DEL BENE? (Si immette)
QUAL È LA DURATA DEL BENE IN ANNI? (Si immette)

FINE	QUOTA	AMMORT	VALORE
ANNO	AMMORT		CORRENTE
			DEL BENE

(Il computer visualizza la tabella)

Le prime righe del programma possono essere scritte senza alcuna spiegazione:

```
100 PRINT "QUAL E' IL VALORE INIZIALE DEL BENE ";
110 INPUT P
120 PRINT "QUAL E' LA DURATA DEL BENE IN ANNI ";
130 INPUT N
140 PRINT
150 PRINT "FINE ","QUOTA ","AMMORT ", "VALORE"
160 PRINT "ANNO","AMMORT";TAB(30);"CORRENTE"
170 PRINT TAB(30);"DEL BENE"
```

Poi dobbiamo calcolare la somma delle cifre degli anni.

```
180 LET S=0
190 FOR I=1 TO N
200 LET S=S+I
210 NEXT I
```

Ora calcoliamo lo schema e stampiamolo. Useremo la variabile P1 per seguire il valore corrente dei beni.

```
220 LET P1=P
230 FOR I=1 TO N
240 LET F=(N+1-I)/S
250 LET D=P*F
260 LET P1=P1-D
270 PRINT I,INT(F*100)/100,INT(D/100)*100,
    INT(P1/100)*100
280 NEXT I
```

Nella riga 240, F è la quota di ammortamento per l'anno I. Potete controllarlo per vari valori di I per assicurarvi che l'espressione generi il valore esatto di F. Nella riga 250, D è l'ammortamento. Nella riga 270 i valori di D e P1 sono arrotondati alle centinaia più vicine. Una approfondita discussione sul metodo migliore di arrotondamento si trova nel primo esempio di programma del Capitolo 9. Ora l'unica cosa che manca è l'istruzione END.

```
290 END
```

Il programma completo è

```
100 PRINT "QUAL E' IL VALORE INIZIALE DEL BENE ";
110 INPUT P
120 PRINT "QUAL E' LA DURATA DEL BENE IN ANNI ";
130 INPUT N
140 PRINT
150 PRINT "FINE ","QUOTA ","AMMORT ", "VALORE"
160 PRINT "ANNO","AMMORT";TAB(30);"CORRENTE"
170 PRINT TAB(30);"BENE"
180 LET S=0
190 FOR I=1 TO N
200 LET S=S+I
210 NEXT I
220 LET P1=P
230 FOR I=1 TO N
240 LET F=(N+1-I)/S
250 LET D=P*F
260 LET P1=P1-D
270 PRINT I,INT(F*100)/100,INT(D/100)*100,
    INT(P1/100)*100
280 NEXT I
290 END
```

6.5 Problemi

1. Scrivete un programma per generare una tabella di numeri e le loro radici quadrate. La tabella dovrebbe apparire così:

```
N        SQR(N)
2.0      1.414214356
2.2      1.4832397
2.4      1.5491933
```

   (ecc.)

2. Scrivete un programma che conti da 0 a 500 per decine e che scriva i risultati.

3. Scrivete un programma che accetti in ingresso un numero N e poi scriva i numeri pari che sono maggiori di zero, ma minori o uguali a N.

4. Scrivete un programma per stampare una tabella di conversione da pollici a centimetri, sapendo che un pollice equivale a 2.54 centimetri. Includete anche le opportune intestazioni. Iniziate la tabella a 0 e continuate fino a 10 pollici con passi di mezzo pollice.

5. Che cosa sarebbe stampato se il programma seguente fosse avviato?

```
100 FOR X=10 TO 1 STEP -1
110 PRINT TAB(X); "ABCDEFGHIJ"
120 NEXT X
130 END
```

6. Studiate il programma seguente. Che cosa si avrà in uscita se fosse eseguito?

```
100 FOR I=1 TO 5
110 READ A
120 LET B=INT(A)-SGN(A)*2
130 PRINT B
140 NEXT I
150 DATA 2,2,-3,10,0,-1.5
160 END
```

7. Che cosa sarà stampato se eseguiamo il seguente programma?

```
100 FOR X=1 TO 10
110 LET Y=2*X
120 FOR Z=1 TO 5
130 LET U=Z + Y
140 FOR V=1 TO 3
150 PRINT V + U
160 NEXT Z
170 NEXT V
180 NEXT X
190 END
```

8. Il programma seguente non funziona. Che cosa c'è che non va?

```
100 FOR X=-10 TO +10 STEP 2
110 PRINT X, SQR(X)
120 NEXT X
130 END
```

9. Spiegate che cosa fa il programma seguente:

```
100 FOR X=1 TO 5
110 READ Y
120 LET Z=INT(100*Y+.5)/100
130 PRINT Z
140 NEXT X
150 DATA 1.06142,27.5292,138.021
160 DATA .423715,51.9132
170 END
```

10. Scrivete un programma per stampare la seguente figura di asterischi:

```
* * * * * * *
  * * * * *
    * * *
      *
```

Non si devono usare più di due istruzioni PRINT!

11. Spiegate cosa fa il seguente programma.

```
100 LET A=-4
110 LET B=ABS(A)
120 LET C=SQR(B)
130 LET D=SGN(A)+C
140 PRINT D
150 END
```

12. N! si legge "N fattoriale" ed equivale al prodotto di tutti i numeri compresi tra 1 e N (compreso). Per esempio

$3! = (1)(2)(3) = 6$
$5! = (1)(2)(3)(4)(5) = 120$

e così via. Scrivete un programma che richieda l'input N, quindi calcoli e scriva N!. Se provate questo programma sul computer potete essere sorpresi nel trovare che ci sono valori di N inferiori a 100 che producono fattoriali troppo grandi per poter essere trattati dal computer. Il fattoriale di N è una funzione che cresce in modo estremamente rapido.

13. Scrivete un programma BASIC che richieda l'immissione di N voti.

Calcolare e stampare il voto più alto, il voto più basso e la media dei voti.

14. Nel seguente programma c'è qualcosa di sbagliato? Se sì, che cosa?

```
100 FOR X=1 TO 2
110 FOR Y=2 TO 6
120 PRINT X+Y
130 NEXT Y
140 FOR Z=1 TO 3
150 PRINT X+Z
160 NEXT X
170 NEXT Z
180 END
```

15. Che cosa si avrà in uscita se verrà eseguito il programma seguente?

```
100 FOR X=1 TO 4
110 FOR Y=1 TO 3
120 LET Z=X*Y
130 PRINT Z;
140 NEXT Y
150 PRINT
160 NEXT X
170 END
```

16. Supponete di decidere di investire L 1000000 il primo dell'anno per 10 anni al tasso di interesse annuo del 6 per cento. Alla fine del decimo anno il valore dell'investimento sarà di L 13971640. Per vedere come questo possa essere calcolato, usate la seguente formula:

CNUOVO = (CVECCHIO + I(1 + R/100).

In questa formula, R è il tasso di interesse annuo in percentuale, I è l'investimento annuo, CVECCHIO è il valore dell'investimento all'inizio di ciascun anno, e CNUOVO è il valore dell'investimento alla fine dell'anno. Così, CNUOVO diventa CVECCHIO per l'anno successivo. Scrivere un programma che produca l'uscita seguente quando viene eseguito:

INVESTIMENTO ANNUALE? (Si immette I)
INTERESSE ANNUALE (%)? (Si immette R)
QUANTI ANNI? (Si immettono gli anni)
ALLA FINE DELL'ULTIMO ANNO IL VALORE
DELL'INVESTIMENTO SARÀ (Il computer scrive la risposta)

17. Le istruzioni DATA qui sotto contengono le ore di lavoro effettuate da un certo numero di impiegati durante il periodo di una settimana.

```
190 DATA 5
200 DATA 2,4800,8,10,8,7,10
201 DATA 5,3750,7,8,8,6,10
202 DATA 1,3250,8,10,6,8,8
203 DATA 4,5000,8,10,6,10,6
204 DATA 3,4250,6,6,8,10,7
```

Il numero della riga 190 dà il numero degli impiegati da seguire. Ciascuna delle righe DATA dopo la riga 190 contiene le registrazioni settimanali per un impiegato. I dati sono il numero di matricola dell'impiegato, il salario orario e le ore di lavoro effettuate da lunedì a venerdì. L'impiegato riceve una volta e mezza la paga oraria per tutte le ore eccedenti le 40 ore settimanali. Scrivere un programma BASIC usando queste istruzioni DATA per calcolare e stampare il numero di matricola dell'impiegato e la paga lorda settimanale per ciascuno degli impiegati.

18. Supponiamo che le seguenti istruzioni DATA diano le votazioni degli studenti di un corso di inglese in tre esami:

```
190 DATA 9
200 DATA 3, 90, 85, 92
201 DATA 1, 75, 80, 71
202 DATA 6, 100, 82, 81
203 DATA 5, 40, 55, 43
204 DATA 2, 60, 71, 68
205 DATA 4, 38, 47, 42
```

Il numero della riga 190 è il numero di studenti del corso. Ciascuna delle istruzioni DATA che seguono dà il profitto per un singolo studente. Le informazioni sono: il numero della tessera di riconoscimento dello studente, voto 1, voto 2, voto 3. Come si vede dalla riga 202, lo studente 6 ha ricevuto i voti d'esame (in centesimi): 100, 82 e 81. Scrivere un programma che usi queste istruzioni DATA per calcolare e stampare per ciascun studente il numero della tessera di riconoscimento e il voto del corso. Si presume che i voti dei primi due esami contino per il 25 per cento ciascuno rispetto al voto finale e che l'ultimo voto conti per il 50 per cento.

6.6 Test di apprendimento

1. Che cosa verrà stampato se sarà eseguito il seguente programma?

```
100 FOR Y=20 TO 1 STEP -2
110 PRINT Y;
120 NEXT Y
130 END
```

....

2. Che cosa verrà stampato se verrà eseguito il programma seguente?

```
100 FOR A=1 TO 4
110 FOR B=1 TO 3
120 PRINT A*B;
130 NEXT B
140 NEXT A
150 END
```

....

3. Completate gli spazi lasciati in bianco.

   a. SQR(36) = _______

   b. INT(7.13) = _______

   c. ABS(−22.8) = _______

   d. SGN(−1.3) = _______

4. C'è qualcosa che non va nel programma seguente? Se sì, che cosa?

```
100 FOR I=1 TO 5
110 FOR J=2 TO 5
120 PRINT I, J
130 NEXT I
140 NEXT J
150 END
```

....

5. Le miglia possono essere convertite in chilometri moltiplicando il numero di miglia per 1.609. Scrivere un programma che produca la seguente tabella quando viene eseguito.

```
MIGLIA        CHILOMETRI
------        ----------
10            16.09
20            32.18
30
(ecc.)
100           160.9
```

....

6. Delle informazioni numeriche sono accumulate in istruzioni DATA nel modo seguente:

```
100 DATA 10
110 DATA 25,21,24,21,26,27,25,24,23,24
```

Il numero della riga 100 indica quanti numeri devono essere elaborati nel resto delle istruzioni DATA. Scrivere un programma usando queste istruzioni per calcolare la media dei numeri escludendo quello contenuto nella riga 100.

....

Capitolo

Lavorare con gli array

7.1 Obiettivi

In questo capitolo applicheremo alle serie di numeri alcune delle idee apprese prima. Saranno introdotti nuovi concetti che espanderanno la vostra capacità di programmare in BASIC. Gli obiettivi sono i seguenti.

VARIABILI CON INDICI SEMPLICI E DOPPI

Vedremo che cosa sono le variabili con indici e come si usano per scrivere programmi potenti.

RISERVARE SPAZIO PER GLI ARRAY

Prima che voi immettiate una serie di numeri nel calcolatore, il calcolatore deve conoscerne le dimensioni. Imparerete a usare l'istruzione DIM per definire la grandezza dell'array.

VARIABILI CON INDICI E ITERAZIONI FOR NEXT

Quando si usano serie di numeri in un programma, quasi certamente il programma dovrà essere ripetitivo. Scoprirete come il loop FOR NEXT può aiutarvi a gestire gli array.

ESEMPI DI PROGRAMMI

Studieremo dei programmi BASIC che mettono a profitto la potenza delle variabili con indici (array).

7.2 Esercizi di scoperta

Poiché lavoreremo con serie di numeri, dobbiamo aggiungere al nostro vocabolario due termini importanti. Potremmo usare la parola serie per descrivere un gruppo di numeri, ma di fatto altre due parole sono più usate: matrice e vettore. Per i nostri scopi entrambe significano la stessa cosa: una "serie di numeri" o array.
Quando lavorate con gli array dovete essere in grado di distinguere un numero in un array dagli altri numeri. Gli indici vi aiutano a distinguere i numeri in un array o in una matrice.

MATRICE e ARRAY significano "serie di numeri"

Per vedere come funziona, diamo uno sguardo all'array qui sotto esposto.

$$Y_1 = 9$$
$$Y_2 = 10$$
$$Y_3 = 7$$
$$Y_4 = 14$$
$$Y_5 = 12$$
$$Y_6 = 15$$

Il nome dell'array è Y. La sua dimensione è 6, dal momento che in esso ci sono 6 elementi (o numeri). I numeri 9, 10, 7, 14, 12 e 15 sono gli elementi dell'array. I numeri stampati a destra e leggermente più in basso delle Y sono chiamati indici: ciascun indice semplicemente punta ad un elemento dell'array. Così Y(4) significa il quarto numero dell'array, che in questo caso è 14. Leggiamo Y(4) come "Y di quattro". Il terzo numero dell'array sarebbe chiamato "Y di tre", e così via. Questo array è monodimensionale, giacché usa un solo numero (o indice) per localizzare un certo elemento dell'array.
Osserviamo ora un esempio un po' più complicato:

$$Z_{1,1} = 4 \quad Z_{1,2} = 9 \quad Z_{1,3} = 5$$
$$Z_{2,1} = 3 \quad Z_{2,2} = 8 \quad Z_{2,3} = 7$$

In questo esempio ci sono sei elementi nell'array Z. Questo però è un ar-

ray bidimensionale, dal momento che dobbiamo specificare quale riga e colonna vogliamo. Il primo indice ci dà il numero di riga; il secondo specifica la colonna. Z(2,1) si legge "Z di due uno" e significa l'elemento di Z alla seconda riga della prima colonna. Parimenti, l'elemento alla riga 1, colonna 3, sarebbe identificato come Z(1,3) e sarebbe letto "Z di uno tre". Riassumendo, lavoreremo con due tipi di matrice o array. L'array monodimensionale (noto anche come "vettore" in italiano) ha bisogno di un solo numero per individuare un elemento dell'array. L'array bidimensionale ha bisogno di due numeri (un numero di riga e un numero di colonna) per localizzare un elemento.

1. Accendete il computer e la TV e immettete il seguente programma:

```
100 LET X(1)=21
110 LET X(2)=13
120 LET X(3)=16
130 LET X(4)=8
140 LET X(5)=11
150 PRINT X(1)
160 END
```

Che cosa pensate che verrà scritto se il programma viene eseguito?

....

Eseguite il programma e segnate quello che è successo.

....

2. Ora modificate il programma per scrivere il quarto valore di X. Eseguite il programma. Ha funzionato?

....

3. Bene, battete

```
150 PRINT X(3) + X(4)
```

Visualizzate il programma e studiatelo brevemente. Che cosa pensate che succeda se eseguite il programma?

....

Eseguite il programma e vedete se avevate ragione. Annotate qui sotto quello che in realtà è stato scritto.

....

4. Battete

```
150 FOR I=1 TO 5
152 PRINT X(I)
154 NEXT I
```

Visualizzate il programma. Che cosa pensate che verrà scritto da questo programma?

....

Vedete se avevate ragione. Registrate qui sotto quello che è successo quando avete eseguito il programma.

....

5. Modificate questo programma per far stampare solo i primi tre valori dell'array X. Annotate qui sotto quanto è successo quando avete provato a fare questo.

....

6. Cancellate il programma in memoria. Immettete il seguente programma:

```
100 LET Y(1,1)=2
110 LET Y(1,2)=5
120 LET Y(1,3)=1
130 LET Y(2,1)=2
140 LET Y(2,2)=4
150 LET Y(2,3)=3
160 PRINT Y(1,3)
170 END
```

L'array creato da questo programma ha le seguenti righe e colonne:

$$\begin{bmatrix} 2 & 5 & 1 \\ 2 & 4 & 3 \end{bmatrix}$$

Visualizzate il programma e assicuratevi di averlo immesso in modo esatto. Che cosa pensate che faccia questo programma?

....

Eseguite il programma e annotate quello che è stato scritto.

....

8. Battere

```
160 PRINT Y(2,2) + Y(1,3) + Y(1,1)
```

Visualizzate il programma. Che cosa farà questo programma quando verrà eseguito?

....

Avviate il programma e vedete se avevate ragione.

9. Battere

```
160 LET S=0
162 FOR J=1 TO 3
164 LET S=S+Y(1,J)
166 NEXT J
168 PRINT S
```

Visualizzate il programma e studiatelo attentamente. Che cosa accadrà se il programma viene eseguito?

....

Eseguite il programma e registrate qui sotto quello che è stato visualizzato.

....

Spiegate con parole vostre quello che avviene nel programma.

....

10. Battete

```
162 FOR I=1 TO 2
164 LET S=S+Y(I,2)
166 NEXT I
```

Visualizzate il programma. Che cosa fa ora?

....

Eseguite il programma e scrivete quello che è stato stampato.

....

Ancora una volta cercate di spiegare con parole vostre quello che succede.

....

11. Ora battete

```
164 FOR J=1 TO 3
166 LET S=S+Y(I,J)
168 NEXT J
170 NEXT I
172 PRINT S
180 END
```

Visualizzate il programma e pensateci su un minuto. In particolare, confrontate quello che vedete ora con quello che succedeva nei punti 9 e 10. Che cosa fa questo programma?

....

Eseguite il programma e annotate quello che è stato scritto.

....

12. Cancellate il programma dalla memoria. Battete il programma seguente:

```
100 DIM X(12), Y(12)
110 FOR I=1 TO 12
120 READ X(I), Y(I)
130 NEXT I
140 PRINT X(1) + Y(4)
150 DATA 2, 1
151 DATA -1, 3
152 DATA 5, 6
153 DATA 2, 4
154 DATA 3, 1
155 DATA 8, 4
156 DATA 5, 1
157 DATA 3, 4
158 DATA 6, 2
159 DATA 1, 1
160 DATA 7, 7
161 DATA 5, 3
170 END
```

Visualizzate il programma e controllate di averlo immesso giusto. Studiatelo attentamente. Se eseguite il programma, che cosa verrà scritto?

....

Eseguite il programma e vedete se avevate ragione o no. Annotate qui sotto quello che è stato scritto.

....

13. Battere

```
100
```

Ora visualizzate le prime righe del programma battendo LIST 100-150. Che cosa è successo?

....

Eseguite il programma e annotate quello che è successo.

....

Appare un BAD SUBSCRIPT ERROR in 110. Vi pare che l'istruzione DIM, originariamente presente nel programma, sia necessaria?

....

14. Battete in modo diretto l'istruzione

```
PRINT I
```

Qual è stato l'indice (I) più grande che il computer ha accettato?

....

15. Battere

```
100 DIM X(9), Y(9)
110 FOR I=1 TO 9
```

Visualizzate il programma. Che succederà ora se avviate il programma?

....

Provatelo e vedete un po' se avevate ragione.

....

16. Battete

```
100
```

Ora che l'istruzione DIM è stata tolta il programma funzionerà ugualmente?

....

Provatelo e registratene qui sotto l'uscita.

....

Confrontate i risultati del punto 13 con quelli del punto 16. Qualche volta l'istruzione DIM ci dev'essere e altre volte non è necessario che ci sia. Torneremo sull'argomento più tardi.

....

17. Cancellate il programma in memoria. Battete il seguente programma:

```
100 DIM A(4,3)
110 FOR I=1 TO 4
120 FOR J=1 TO 3
130 READ A(I,J)
140 NEXT J
150 NEXT I
160 FOR I=1 TO 4
170 FOR J=1 TO 3
180 PRINT A(I,J);
190 NEXT J
200 PRINT
210 PRINT
220 NEXT I
230 DATA 1,3,1
240 DATA 4,2,5
250 DATA 1,4,2
260 DATA 3,2,5
270 END
```

Assicuratevi di avere immesso correttamente il programma, poi stu-

diatelo per qualche minuto. Riuscite a immaginare quello che verrà scritto?

....

Eseguite il programma e annotatene l'uscita.

....

Confrontate quello che è stato stampato con i numeri delle istruzioni DATA nel programma.

18. Cancellate il programma dalla memoria e immettete poi il programma seguente:

```
100 DIM A(2,2)
110 FOR I=1 TO 2
120 INPUT A(I,1),A(I,2)
130 NEXT I
140 PRINT
150 PRINT
160 FOR I=1 TO 2
170 FOR J=1 TO 2
180 PRINT A(I,J);
190 NEXT J
200 PRINT
210 NEXT I
220 END
```

Eseguite il programma e quando compare il segnale di input, battete

2,5
3,8

Che cosa è successo?

....

Confrontate l'uscita con i numeri che avete immesso.

19. Cancellate il programma in memoria. Poi immettete il programma seguente:

```
100 DIM X(3,3)
110 FOR I=1 TO 3
```

```
120 FOR J=1 TO 3
130 READ X(I,J)
140 NEXT J
150 NEXT I
160 PRINT
170 PRINT
180 FOR I=1 TO 3
190 FOR J=1 TO 3
200 PRINT X(I,J);
210 NEXT J
220 PRINT
230 NEXT I
240 DATA 2,1,3
250 DATA 4,7,5
260 DATA 1,2,6
270 END
```

Eseguite il programma. Che cosa è successo?

....

Confrontate l'uscita con i numeri contenuti nelle istruzioni DATA.

20. Con ciò si conclude per questo capitolo il lavoro sul computer. Spegnete il computer e la TV.

7.3 Analisi

A questo punto potreste avere idee un po' confuse a proposito degli array. Il materiale di questa analisi dovrebbe chiarire qualunque dubbio che sia potuto sorgere durante il lavoro sul computer.

VARIABILI CON INDICI SEMPLICI E DOPPI

La necessità di avere dei numeri con indici diventa evidente quando dobbiamo trattare grandi raccolte di numeri. Se, per esempio, stessimo scrivendo un programma che coinvolge solo quattro numeri, non avremmo difficoltà nel dar loro un nome. Potremmo chiamare i numeri X, Y, U e V. Ma se si dovesse lavorare con 100 numeri? Per questo motivo è spesso molto utile avere dei numeri con indici. Fortunatamente il BASIC ha la possibilità di avere gli indici, pronti all'uso.

Considerate il seguente gruppo di numeri:

I	Y_I
1	14
2	8
3	9
4	11
5	16
6	20
7	5
8	3

Possiamo riferirci all'intero gruppo di numeri con l'unico nome Y. Così, Y è una raccolta di numeri, una matrice, o un array, significando questi tre nomi approssimativamente la stessa cosa per i nostri scopi. Per individuare un numero in un array, dobbiamo avere il nome dell'array (in questo caso Y) e la posizione nell'array. Qui è dove viene usata la colonna I. Così Y(3) (che leggiamo "Y di tre") individua il terzo numero nell'array Y. In questo caso, Y(3) ha il valore di 9. Similmente, Y(7) è 5, Y(1) è 14, e così via. Possiamo assegnare le norme Y(I) (che sarebbe pronunciato "Y di I"), a qualunque elemento dell'array dipendente dal valore di I. Se I fosse 7, allora Y(I) sarebbe 5 nel nostro esempio. Questa serie di numeri è monodimensionale, giacché solo un numero (indice) è necessario per localizzare un qualsiasi elemento nell'array.
Ora diamo uno sguardo ad un array bidimensionale.

$Y_{I,J}$	1	2	3	4
1	3	−1	10	8
2	2	4	5	6
3	1	−2	9	3

Ora ci servono due numeri per localizzare un elemento nell'array. Dato un numero di riga e un numero di colonna, possiamo trovare qualunque elemento vogliamo nell'array. Per esempio, Y(1,3) significa l'elemento di Y situato alla riga 1, colonna 3. Nell'esempio suddetto l'elemento ha il valore 10. In generale, si denota un elemento in un array bidimensionale come Y(I,J). Il primo indice (I) è il numero di riga, e il secondo indice (J) è il numero di colonna.

I doppi indici definiscono i numeri di riga e di colonna

Per essere sicuri che comprendiate come sono usati i doppi indici, fate riferimento all'array bidimensionale della tabella riportata sopra e verifi-

cate che le affermazioni seguenti siano corrette:

$$Y_{3,2} = -2$$
$$Y_{1,4} = 8$$
$$Y_{3,3} = 9$$
$$Y_{2,1} = 2$$

Il computer può interpretare gli indici in forma di espressioni così X(M−N+3,S*T) è corretto, ammesso che il computer possa convertire M−N+3 e S*T in numeri. Persino Y(Y(1,1),Y(2,3)) va bene fin tanto che il computer può localizzare i numeri in Y(1,1) e Y(2,3). C'è però un punto importante da ricordare. Supponiamo di voler vedere X(A+B) dove A = 2.6 e B = 1.1. Allora A+B = 3.7. Ma non ha alcun senso cercare di vedere il 3.7° numero nell'array X. In questo caso, il computer prenderebbe la parte intera di 3.7 e X(A+B) risulterebbe essere X(3), il quarto elemento dell'array X.

RISERVARE SPAZIO PER GLI ARRAY

Il computer deve sapere quant'è grande una matrice per due motivi: primo, c'è la questione di quanto spazio riservare in memoria per contenere la matrice. Poi, il computer deve conoscere la dimensione della matrice per poter effettuare in modo appropriato le operazioni aritmetiche. In realtà, per le matrici piccole, il BASIC riserva spazio automaticamente. Se in un programma viene usata una matrice monodimensionale, il BASIC riserva automaticamente spazio per dieci elementi, se non vi è un'istruzione DIM. Se viene usata una matrice bidimensionale, riserverà spazio in memoria abbastanza per una matrice di dieci per dieci, sempre se non vi è nel programma alcuna istruzione DIM. È forse poco saggio usare questa caratteristica del BASIC. In questo libro metteremo l'accento sull'uso abituale di istruzioni di dimensionamento in tutti i programmi, indipendentemente dalle dimensioni delle matrici.

Riservare spazio con un'istruzione DIM

Un esempio di istruzione DIM (che sta per "dimensione") è

```
100 DIM B(5,20),Y(8),Z(34),X(3,6)
```

In questa istruzione sono dimensionati quattro array. B è una matrice bidimensionale con cinque righe e venti colonne. Y è un array monodimensionale con otto elementi; Z è monodimensionale, con trentaquattro elementi. Infine X è una matrice bidimensionale con tre righe e sei colonne.

L'istruzione DIM deve venire prima di qualunque altra istruzione che si riferisca agli array. Come indicato sopra, è buona pratica usare un'istruzione DIM in tutti i programmi. Dovrete perciò dare un'occhiata all'inizio del programma per vedere le misure degli array che saranno usati.

VARIABILI CON INDICI E ITERAZIONI FOR NEXT

Dal momento che gli indici individuano serie di numeri e le operazioni con serie di numeri quasi sempre implicano ripetizioni, sembra ragionevole usare le istruzioni FOR NEXT per trattare le matrici. Come esempio, il seguente segmento di programma costituisce un array 6 per 4, poi carica dei cinque in tutti gli elementi.

```
100 DIM A(6,4)
110 FOR R=1 TO 6
120 FOR C=1 TO 4
130 LET A(R,C)=5
140 NEXT C
150 NEXT R
```

Se studiamo questo segmento di programma, i dettagli del processo diventano chiari. Quando la riga 130 del programma viene raggiunta la prima volta, R = 1 e C = 1. Quindi R viene tenuto costante mentre C va a 2, 3 e 4. Ad ogni passo nel processo, il corrispondente elemento della matrice è posto uguale a 5. Poi R è posto uguale a 2, e C prende i valori 1, 2, 3 e 4. Il processo continua fino a che tutti gli elementi dell'array non siano stati resi uguali a 5.
Sia gli array ad una che quelli a due dimensioni possono essere trattati in questo modo usando gli indici. In molte applicazioni è preferibile usare le iterazioni FOR NEXT per eseguire le operazioni desiderate sulle matrici.

7.4 Esempi di programmi

ESEMPIO 1 — VOTI D'ESAME

Per illustrare il concetto di array monodimensionale (vettore), prendiamo ad esempio un gruppo di voti di esame. I seguenti sono i risultati di un esame sostenuto da una classe di 15 studenti:

Numero studente	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Voto	6,5	8	9,5	7,5	4,5	6,5	9	9	7,5	7	6,5	8	7,5	7	7

Il problema è scrivere un programma BASIC per permettere l'immissione dei suddetti voti. Il formato è:

QUANTI STUDENTI? (Si immette il numero)
STUDENTE VOTO
1 (Si batte il voto)
2 (Si batte il voto)
3 (Si batte il voto)
(ecc.) (ecc.)

Il programma deve calcolare la media della classe, il voto più alto e il voto più basso, e stampare queste informazioni come segue:

LA MEDIA DELLA CLASSE È (Il computer scrive la media)
IL VOTO PIÙ ALTO È (Il computer scrive il voto più alto)
IL VOTO PIÙ BASSO È (Il computer scrive il voto più basso)

Come negli esercizi precedenti, affrontiamo il problema per gradi. Primo, siccome memorizzeremo i voti degli studenti con degli indici, dobbiamo mettere un'istruzione DIM per risparmiare spazio per l'array.

```
100 DIM V(50)
```

Usiamo la variabile V per memorizzare i voti: possiamo inserirvi fino a cinquanta voti. Poi avremo un messaggio, un'immissione e uno spazio.

```
110 PRINT "QUANTI STUDENTI";
120 INPUT N
130 PRINT
```

Per prima cosa deve essere generata l'intestazione della tabella.

```
140 PRINT "STUDENTE", "VOTO"
150 PRINT
```

Ora siamo pronti per immettere i voti. Un'iterazione che usi le istruzioni FOR NEXT è l'ideale per controllare l'immissione dei voti.

```
160 FOR I=1 TO N
170 PRINT I,
180 INPUT V(I)
190 NEXT I
```

Il numero dello studente viene emesso nella riga 170. Nella riga 180 il numero dello studente (I) viene usato come indice per il voto. In tal modo vengono generati nel computer i voti nella forma V(1), V(2),..., V(n). Il compito successivo è quello di trovare la media dei voti. Questo si può fare facendo la somma di tutti i voti e dividendo per il numero dei voti.

```
200 LET S=0
210 FOR I=1 TO N
220 LET S=S+V(I)
230 NEXT I
240 LET M=S/N
```

Ora possiamo calcolare la media e far scrivere il risultato.

```
250 PRINT
260 PRINT "LA MEDIA DELLA CLASSE E'";M
```

La parte finale del programma individua e scrive il più alto e il più basso dei voti della classe. A e B staranno rispettivamente per il più alto e il più basso dei voti. Inizialmente renderemo sia A che B uguali a V(1), il primo voto della lista. Sappiamo che lo stesso voto non potrà essere nello stesso tempo il più alto e il più basso. Così, faremo passare il resto dei voti, confrontando A e B con ciascun voto, e faremo ad A e B gli aggiustamenti necessari a seconda delle necessità.

```
270 LET A=V(1)
280 LET B=V(1)
290 FOR I=2 TO N
300 IF B<V(I) THEN 320
310 LET B=V(I)
320 IF A>V(I) THEN 340
330 LET A=V(I)
340 NEXT I
```

L'uscita dei dati si può ottenere con due righe.

```
350 PRINT "IL VOTO PIU' ALTO E'";A
360 PRINT "IL VOTO PIU' BASSO E'";B
```

Infine l'istruzione END completa il programma.

```
370 END
```

Segue il programma completo:

```
100 DIM V(50)
110 PRINT "QUANTI STUDENTI";
120 INPUT N
130 PRINT
140 PRINT "STUDENTE", "VOTO"
150 PRINT
160 FOR I=1 TO N
170 PRINT I,
180 INPUT V(I)
190 NEXT I
200 LET S=0
210 FOR I=1 TO N
220 LET S=S+V(I)
230 NEXT I
240 LET M=S/N
250 PRINT
260 PRINT "LA MEDIA DELLA CLASSE E'";M
270 LET A=V(1)
280 LET B=V(1)
290 FOR I=2 TO N
300 IF B<V(I) THEN 320
310 LET B=V(I)
320 IF A>V(I) THEN 340
330 LET A=V(I)
340 NEXT I
350 PRINT "IL VOTO PIU' ALTO E'";A
360 PRINT "IL VOTO PIU' BASSO E'";B
370 END
```

Accendete il computer e la TV, ed eseguite questo programma usando i dati riportati all'inizio. Se avete qualche difficoltà con le ricerche del voto più alto e del più basso nelle righe da 270 a 340, esaminate in dettaglio il programma.

ESEMPIO 2 — VOTI DEL CORSO

Possiamo facilmente estendere le idee dell'esempio 1 ad un array bidimensionale. Supponiamo ora di avere una classe di dieci studenti e che il voto finale si basi su cinque prove d'esame con votazione in centesimi. Dei risultati tipici per una classe di questo tipo potrebbero essere

		Numero studente									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Esame	**1**	92	71	81	52	75	97	100	63	41	75
	2	85	63	79	49	71	91	93	58	52	71
	3	89	74	80	61	79	88	97	55	51	73
	4	96	68	84	58	80	93	95	61	47	70
	5	81	72	82	63	73	92	93	68	56	74

Usiamo una matrice con l'istruzione FOR NEXT per leggere (READ) da un'istruzione DATA. Il computer deve calcolare ed emettere le seguenti informazioni:

STUDENTE MEDIA DEL CORSO
1 (Il computer stampa la media, ecc.)
2
3
ecc.
ESAME MEDIA DELLA CLASSE
1 (Il computer stampa la media, ecc.)
2
3
ecc.

Il programma deve iniziare con un'istruzione DIM.

```
100 DIM G(5,10)
```

Questa istruzione riserva spazio in memoria per un array con 5 righe e 10 colonne. Il numero di riga (R) sarà il numero dell'esame, e il numero di colonna (C) corrisponderà al numero dello studente. Inseriamo le istruzioni DATA a questo punto:

```
110 DATA 92,71,81,52,75,97,100,63,41,75
120 DATA 85,63,79,49,71,91,93,58,52,71
130 DATA 89,74,80,61,79,88,97,55,51,73
140 DATA 96,68,84,58,80,93,95,61,47,70
150 DATA 81,72,82,63,73,92,93,68,56,74
```

Le seguenti istruzioni FOR NEXT, con l'istruzione READ alla riga 180, rendono disponibili tutti i dati al computer:

```
160 FOR R=1 TO 5
170 FOR C=1 TO 10
```

```
180 READ G(R,C)
190 NEXT C
200 NEXT R
```

Questo fa sì che i numeri vengano letti e immessi nella matrice G per righe. Così, i dati della riga 110 diventano la riga 1 della matrice G, e così via. Prima di fare qualunque altra cosa, dobbiamo far stampare le necessarie intestazioni.

```
210 PRINT "STUDENTE","MEDIA DEL CORSO"
220 PRINT
```

Ora possiamo calcolare la media del corso per ciascuno studente.

```
230 FOR C=1 TO 10
```

La riga 230 apre un'iterazione che va a vedere ogni colonna della matrice. Per ciascun valore di C, dobbiamo calcolare la media della colonna e farne scrivere il risultato.

```
240 LET S=0
250 FOR R=1 TO 5
260 LET S=S + G(R,C)
270 NEXT R
280 PRINT C,S/5
```

Poi chiudere l'iterazione C.

```
290 NEXT C
```

Ora il procedimento viene ripetuto, con la differenza che adesso le medie sono calcolate per righe, invece che per colonne. I risultati rifletteranno la media della classe in ciascun esame.

```
300 PRINT
310 PRINT "ESAME","MEDIA DELLA CLASSE"
320 PRINT
330 FOR R=1 TO 5
340 LET S=0
350 FOR C=1 TO 10
360 LET S=S + G(R,C)
370 NEXT C
380 PRINT R,S/10
390 NEXT R
```

E infine l'istruzione END:

```
400 END
```

Il programma completo è elencato qui sotto:

```
100 DIM G(5,10)
110 DATA 92,71,81,52,75,97,100,63,41,75
120 DATA 85,63,79,49,71,91,93,58,52,71
130 DATA 89,74,80,61,79,88,97,55,51,73
140 DATA 96,68,84,58,80,93,95,61,47,70
150 DATA 81,72,82,63,73,92,93,68,56,74
160 FOR R=1 TO 5
170 FOR C=1 TO 10
180 READ G(R,C)
190 NEXT C
200 NEXT R
210 PRINT "STUDENTE","MEDIA DEL CORSO"
220 PRINT
230 FOR C=1 TO 10
240 LET S=0
250 FOR R=1 TO 5
260 LET S=S + G(R,C)
270 NEXT R
280 PRINT C,S/5
290 NEXT C
300 PRINT
310 PRINT "ESAME","MEDIA DELLA CLASSE"
320 PRINT
330 FOR R=1 TO 5
340 LET S=0
350 FOR C=1 TO 10
360 LET S=S + G(R,C)
370 NEXT C
380 PRINT R,S/10
390 NEXT R
400 END
```

Questo programma è interessante e illustra importanti tecniche di programmazione con gli array. Vale la pena studiarlo ed eseguirlo sul computer.

ESEMPIO 3 — OPERAZIONI SUGLI ARRAY

Segue una serie di brevi programmi che saranno dati senza spiegazioni. Studiate ciascun programma fino a che non siate sicuri di capire quello che succede.

a. Il programma usando iterazioni FOR NEXT carica una matrice tre per quattro con degli 1.

```
100 DIM X(3,4)
110 FOR R=1 TO 3
120 FOR C=1 TO 4
130 LET X(R,C)=1
140 NEXT C
150 NEXT R
160 END
```

b. Questo programma genera e carica i numeri

2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048

in un array monodimensionale

```
100 DIM Z(11)
110 LET Z(1)=2
120 FOR I=2 TO 11
130 LET Z(I)=2*Z(I-1)
140 NEXT I
150 END
```

c. Il programma legge da istruzioni DATA e poi stampa l'array

$$\begin{bmatrix} 2 & 3 & 5 \\ 1 & 4 & 2 \end{bmatrix}$$

```
100 DIM A(2,3)
110 FOR R=1 TO 2
120 FOR C=1 TO 3
130 READ A(R,C)
140 NEXT C
150 NEXT R
160 FOR R=1 TO 2
170 FOR C=1 TO 3
180 PRINT A(R,C);
190 NEXT C
200 PRINT
210 NEXT R
220 DATA 2,3,5
230 DATA 1,4,2
240 END
```

7.5 Problemi

1. Scrivere un programma usando le istruzioni DATA

```
200 DATA 12
210 DATA 2,1,4,3,2,4,5,6,3,5,4,1
```

che legga la dimensione di un array dalla prima istruzione DATA, quindi legga gli elementi dell'array dalla seconda istruzione DATA, caricandoli in un array X; poi stampi l'array.

2. Scrivere un programma per riempire un array di 3 per 4 con degli 1.

3. Scrivere un programma che richieda l'immissione di una matrice quadrata N per N, dove N è un numero intero non superiore a 10. Il programma deve calcolare e stampare la somma dei dati sulla diagonale principale della matrice (dove il primo e secondo indice sono uguali).

4. Scrivere un programma BASIC usando il comando READ per leggere venticinque numeri dalle istruzioni DATA e metterli in un array monodimensionale chiamato A. Fare una ricerca nell'array e stampare il numero degli elementi dell'array che sono più grandi di cinquanta. Completare le istruzioni DATA con dei numeri a piacere.

5. Scrivere un programma che richieda l'immissione di una matrice M per N. Presumere che sia M che N non siano maggiori di 15. Quindi calcolare e stampare la somma di tutti gli elementi della matrice.

6. Il seguente programma dovrebbe calcolare e scrivere la somma degli elementi di un array monodimensionale che siano positivi, ma non maggiori di 10. Così com'è, il programma è sbagliato. Che cosa c'è che non va?

```
100 DIM A(6)
110 FOR I=1 TO 6
120 INPUT A(I)
130 NEXT I
140 LET S=0
150 FOR I=6 TO 1 STEP -1
160 IF A(I)>10 THEN 180
170 LET S=S+A(I)
180 NEXT I
190 PRINT S
200 END
```

7. Che cosa si avrà in uscita se viene eseguito il programma seguente?

```
100 DIM Y(6)
110 FOR I=1 TO 6
120 READ Y(I)
130 NEXT I
140 DATA 2,1,3,1,2,1
150 LET S1=0
160 LET S2=0
170 FOR I=1 TO 6
180 LET S1=S1 + Y(I)
190 LET S2=S2 + Y(I)↑2
200 NEXT I
210 LET X=S2 - S1
220 PRINT X
230 END
```

8. Che cosa si avrà in uscita quando il seguente programma verrà eseguito?

```
100 DIM A(10)
110 FOR I=1 TO 10
120 READ A(I)
130 NEXT I
140 LET S=A(1)
150 FOR I=1 TO 9
160 LET A(I)=A(I+1)
170 NEXT I
180 LET A(10)=S
190 FOR I=1 TO 10
200 PRINT A(I)
210 NEXT I
220 DATA 10,9,8,7,6,5,4,3,2,1
230 END
```

9. Che cosa verrà stampato se verrà eseguito il seguente programma?

```
100 DIM X(4,4)
110 FOR I=1 TO 4
120 FOR J=1 TO 4
130 READ X(I,J)
140 NEXT J
150 NEXT I
160 DATA 1,2,3,4,2,3,4,5
170 DATA 3,4,5,6,4,5,6,7
180 LET S=0
190 FOR I=1 TO 4
200 LET S=S+X(I,5-I)
```

```
210 NEXT I
220 PRINT S
230 END
```

10. Che cosa verrà stampato se viene eseguito il programma che segue?

```
100 DIM Y(4,4)
110 FOR R=1 TO 4
120 FOR C=1 TO 4
130 LET Y(R,C)=0
140 NEXT C
150 NEXT R
160 FOR R=1 TO 4
170 FORC=1 TO 4
180 LET Y(R,C)=R*C
190 NEXT C
200 NEXT R
210 FOR R=1 TO 4
220 FOR C=1 TO 4
230 PRINT Y(R,C);
240 NEXT C
250 PRINT
260 NEXT R
270 END
```

11. Scrivere un programma BASIC che richieda l'immissione di N (che si presume essere un numero intero fra 1 e 100), quindi generi un array monodimensionale con N elementi. Metta in ordine i numeri dell'array in modo ascendente, e infine stampi l'array ordinato.

12. Poniamo che il primo numero delle istruzioni DATA dia il numero degli elementi che devono seguire. Si presuma che i dati siano tutti dei numeri interi compresi fra 1 e 10 estremi inclusi. Scrivere un programma che calcoli il numero degli 1, dei 2, ecc., nei dati e che quindi lo scriva. (Suggerimento: usare i dati man mano che sono letti come un indice per incrementare un elemento di un array usato per contare i numeri.)

13. Che cosa verrà stampato se il seguente programma viene eseguito?

```
100 DIM Z(6,6)
110 FOR R=1 TO 6
120 FOR C=1 TO 6
130 LET Z(R,C)=0
140 NEXT C
150 NEXT R
160 FOR R=1 TO 5 STEP 2
```

```
170 FOR C=R TO 6
180 LET Z(R,C)=1
190 NEXT C
200 NEXT R
210 FOR R=1 TO 6
220 FOR C=1 TO 6
230 PRINT Z(R,C);
240 NEXT C
250 PRINT
260 NEXT R
270 END
```

14. Se il programma che segue venisse eseguito, che cosa scriverebbe il computer?

```
100 DIM A(5,5)
110 FOR R=1 TO 5
120 FOR C=1 TO 5
130 READ A(R,C)
140 NEXT C
150 NEXT R
160 DATA 2,2,2,2,2,2,2,2,2,2
170 DATA 2,2,2,2,2,2,2,2,2,2
180 DATA 2,2,2,2,2
190 FOR C=5 TO 1 STEP -1
200 FOR R=1 TO C
210 LET A(R,C)=3
220 NEXT R
230 NEXT C
240 FOR R=1 TO 5
250 FOR C=1 TO 5
260 PRINT A(R,C);
270 NEXT C
280 PRINT
290 NEXT R
300 END
```

15. Scrivere un programma per leggere l'array seguente da istruzioni DATA, quindi stampare l'array.

$$\begin{bmatrix} 2 & 1 & 0 & 5 & 1 \\ 3 & 2 & 1 & 3 & 1 \end{bmatrix}$$

16. Scrivere un programma per leggere l'array seguente da istruzioni DATA, quindi stampare l'array.

$$\begin{bmatrix} 5 & 3 \\ 2 & 0 \\ -1 & 1 \\ 4 & 2 \\ 2 & 6 \end{bmatrix}$$

17. Scrivere un programma **BASIC** che richiede l'immissione di una matrice M per N. Quindi calcoli ed emetta la somma degli elementi di ciascuna riga e il prodotto degli elementi di ciascuna colonna.

18. Scrivere un programma BASIC che legga due matrici da istruzioni DATA. Entrambe le matrici sono due per tre. Quindi calcoli una terza matrice due per tre tale che ciascun elemento sia la somma dei corrispondenti elementi delle prime due matrici. Stampi infine la terza matrice.

19. I dati qui sotto rappresentano i totali delle vendite effettuate da degli agenti nel periodo di una settiamana

		Lun	**Mar**	**Mer**	**Gio**	**Ven**	**Sab**
Agente	**1**	48	40	73	120	100	90
	2	75	130	90	140	110	85
	3	50	72	140	125	106	92
	4	108	75	92	152	91	87

Scrivere un programma che calcoli e scriva
a. Il totale delle vendite giornaliere
b. Il totale delle vendite settimanali per agente
c . Il totale delle vendite settimanali

20. Scrivere un programma che richieda l'immissione di una matrice 4 per 4 e calcoli una nuova matrice dalla prima con le righe e le colonne scambiate fra loro. Cioè, la riga 2 della matrice immessa deve diventare la colonna 2 della nuova matrice, e così via. Il programma dovrà poi stampare la nuova matrice.

21. Si considerino i due array seguenti:

P	X
1	28
5	2
3	14

4 3
2 17
6 9

Ciascun elemento di P punta ad un elemento di X. P(1) = 1 e X(1) = 28. P(2) = 5 e X(5) = 17. Se tenete attivo questo processo, i valori di X sono elencati in ordine decrescente. Scrivere un programma che fissi due array X e P ad una lunghezza conveniente, e che richieda poi l'immissione di valori arbitrari di X dalla tastiera. Costruire l'array P in modo che i suoi elementi puntino a X in ordine discendente come illustrato sopra. Quindi stampi i due array come mostrato.

7.6 Test di apprendimento

1. Qual è lo scopo dell'istruzione DIM?

....

2. Abbiamo una matrice chiamata X. Che nome di variabile usa il BASIC per individuare l'elemento che si trova nella riga 3, colonna 4?

....

3. Usare un array in un programma per immettere un elenco di numeri, quindi trovare e stampare la somma dei numeri positivi dell'elenco. L'uscita deve apparire così:

   QUANTI NUMERI? (Si immette il numero)
   QUALI SONO I NUMERI? (Si immettono)
   LA SOMMA DEGLI ELEMENTI POSITIVI È (Il computer visualizza la risposta)

....

4. Scrivere un programma usando le istruzioni FOR NEXT per caricare con dei 4 un array di quattro per sei.

....

5. Che cosa sarà stampato se verrà eseguito il seguente programma?

```
100 DIM A(5,5)
110 FOR I=1 TO 5
120 FOR J=1 TO 5
130 LET A(R,C)=0
140 NEXT J
150 NEXT I
160 FOR I=1 TO 5
170 LET A(I,I)=2
180 NEXT I
190 FOR I=1 TO 5
200 FOR J=1 TO 5
210 PRINT A(I,J);
220 NEXT J
230 PRINT
240 NEXT I
250 END
```

....

6. Il seguente array è chiamato A:

$$\begin{bmatrix} 1 & 3 & 5 \\ 6 & 2 & 4 \end{bmatrix}$$

a. Scrivere un'istruzione DIM per A.

....

b. Qual è il valore di A(2,3)?

....

c. Se X = 1 e Y = 2, che cos'è A(X,Y)?

....

d. Che cos'è A(A(1,1),A(2,2))?

....

Capitolo

Variabili a stringa 8

8.1 Obiettivi

Alcune delle più importanti applicazioni dei computer sono non-numeriche e trattano dei caratteri piuttosto che dei numeri. Le stringhe di caratteri possono essere trattate come "variabili a stringa" e sono l'argomento di questo capitolo. Vedremo in particolare i seguenti argomenti che sono collegati con le stringhe.

INPUT E OUTPUT DI STRINGHE

Prima di poter svolgere delle operazioni significative sulle stringhe, dobbiamo imparare come sono gestite dal computer le operazioni di input e di output delle variabili a stringa.

FUNZIONI SU STRINGHE

Avete già studiato delle funzioni BASIC che operano sui numeri. Ora tratteremo le funzioni che operano su stringhe di caratteri.

ESEMPI DI PROGRAMMA

Lo scopo finale è quello di scrivere dei programmi che funzionino con variabili a stringa.

8.2 Esercizi di scoperta

1. Accendete il computer e la TV. Immettete il seguente programma:

```
100 INPUT A$
110 PRINT
120 PRINT A$
130 PRINT A$
140 END
```

Nel programma A$ identifica la variabile come una variabile a stringa. Eseguite il programma e al segnale di immissione (il punto interrogativo) battete il vostro nome per intero. Che cosa è successo?

....

2. Ora modificate il programma come segue:

```
100 INPUT A$,B$
110 PRINT
120 PRINT B$
130 PRINT A$
140 END
```

Se eseguite il programma e al segnale di immissione battete le parole INTELLIGENTE e CONVERSAZIONE separate da una virgola, che cosa pensate che verrà scritto in risposta dal computer?

....

Provate e annotate quello che è successo.

....

3. Cancellate il programma e immettete il seguente:

```
100 READ X,X$,Y,Y$
110 DATA 10,ROSI,20,CARLO
120 PRINT X,Y
130 PRINT X$,Y$
140 END
```

Questo programma contiene diverse idee nuove. Che cosa pensate che succeda se si esegue il programma?

....

Eseguite il programma e registrate qui sotto quello che è avvenuto.

....

4. Come vedete, potete includere stringhe nelle istruzioni DATA. Cambiate la riga 110 nel modo seguente:

```
110 DATA 10,"ROSI",20,"CARLO"
```

Eseguite il programma. Cosa è successo?

....

5. Cambiate la riga 110 così

```
110 DATA "10","ROSI",20,CARLO
```

Visualizzate il programma ed eseguitelo. Cosa è successo?

....

"10" è una stringa mentre X cerca un numero.

6. Cambiate la riga 100

```
100 READ X$,X,Y,Y$
```

Che cosa succede se si cerca di eseguire il programma in questa forma?

....

Vedete se avevate ragione. Registrate qui sotto quello che è successo quando avete cercato di eseguire il programma.

....

7. L'errore di sintassi è causato da una discrepanza nei tipi di dati. Visualizzate il programma ed osservate che ROSI è una stringa, ma X è una variabile numerica. È chiaro che si deve stare attenti a che le informazioni delle istruzioni DATA vadano d'accordo con il tipo di variabile delle istruzioni READ. Passiamo ora ad un altro argomento. Cancellate il programma dalla memoria del computer e immettete il seguente:

```
100 INPUT C$
110 LET N=LEN(C$)
120 PRINT N
130 END
```

La caratteristica nuova in questo programma è la funzione LEN(C$). È nuova perché lavora su una stringa (in questo caso la stringa C$) invece che su un numero. Potete immaginare quello che fa la funzione?

....

Notate che il risultato della funzione LEN che opera su una stringa dev'essere un numero, dal momento che il risultato viene assegnato ad una variabile numerica N.

8. Eseguite il programma e al segnale di immissione battete ABCDE. Che cosa è stato scritto in risposta?

....

Provate di nuovo, ma questa volta battete VITADACANI. Che cosa è successo?

....

A questo punto dovreste avere un'idea abbastanza precisa di quello che fa la funzione LEN.

9. Eccovi un nuovo problema. Se eseguite il programma e al segnale di immissione premete semplicemente il tasto RETURN che cosa pensate che verrà scritto dal computer in risposta?

....

Provate e controllate se avevate ragione. Ora, ancora una variazione. Eseguite il programma e al segnale di immissione immettete R O - B E R T O. Notate gli spazi fra i caratteri. Che cosa è stato scritto in risposta?

....

Nella funzione LEN gli spazi contano come caratteri?

....

10. Ora vogliamo specificare una sottostringa in una stringa. Cioè, data una stringa A$ come possiamo specificare M caratteri di quella stringa cominciando dall'N-esimo carattere della stringa? La funzione che dà questa sottostringa è MID$(A$,N,M). Dà M caratteri presi da A$ a partire dall'N-esimo carattere.
Ora cancellate il programma in memoria e immettete quello che segue:

```
100 INPUT A$
110 PRINT "M = ";
120 INPUT M
130 PRINT "N = ";
140 INPUT N
150 PRINT MID$(A$,N,M)
160 END
```

Eseguite il programma e al segnale di immissione della stringa, battete MISSISSIPPI. Notate che la lunghezza di questa stringa è 11. Immettete 5 per M e 4 per N. Che cosa è successo?

....

11. Cancellate dalla memoria del computer il programma e immettete questo:

```
100 INPUT A$
110 INPUT J
120 PRINT MID$(A$,J,1)
130 PRINT
140 GOTO 110
150 END
```

Eseguite il programma e alla prima richiesta di immissione, battete ABCDEFGHIJKLMNOPQRSTUVWXYZ. Alla seconda richiesta di immissione battete 20. Che cosa è successo?

....

Il computer è alla richiesta di immissione per J. Questa volta battete 10. Che cosa è successo?

....

Alla terza richiesta di input sperimentate diversi valori di I fra 1 e

26. Descrivete con parole vostre che cosa succede quando il computer è diretto a stampare MID$(A$,I,1).

....

12. Resta da controllare un'ultima cosa. Dovreste ancora trovarvi alla richiesta di input per J. Battete 30. Che cosa è successo?

....

A questo punto dovreste capire in modo abbastanza chiaro che cosa succede quando il computer stampa MID$(A$,J,1). Naturalmente in questo caso il valore di J è più grande della lunghezza della stringa. Torneremo su questo argomento nella sezione Analisi. Fate uscire il computer dall'iterazione.

13. Fate per conto vostro esperimenti con questo programma. Provate varie stringhe e diversi valori di M e N fino a quando non capite esattamente come lavora la funzione MID$.

14. Passiamo ora ad un argomento diverso. Cancellate il programma in memoria e immettete il seguente:

```
100 INPUT A$
110 INPUT B$
120 IF A$ < B$ THEN 150
130 PRINT B$
140 GOTO 100
150 PRINT A$
160 GOTO 100
170 END
```

Studiate per qualche istante il programma. È chiaro che la parte interessante è nella riga 120 dove le stringhe A$ e B$ sono coinvolte in un'istruzione IF THEN. In particolare, che cosa pensate che voglia dire A$<B$ trattandosi di stringhe?

....

Il modo per vedere se avete ragione o no è quello di eseguire il programma e fare alcune prove sul computer. Eseguite il programma e alla prima richiesta di immissione di dati battete OCA; alla seconda richiesta battete GALLINA. Che cosa è stato scritto in risposta?

....

15. Il computer attende che immettiate una stringa. Questa volta battete CASA seguito da TELEVISIONE. Che cosa è stato stampato?

    ....

    Continuate a fare esperimenti con parole o lettere di vostra scelta fino a quando non vediate esattamente che cosa significa l'espressione A$<B$. Una volta capito questo, vi dovrebbe essere facile vedere che cosa potrebbero significare A$ = B$, A$>B$, o A$<>B$.

16. Fate uscire il computer dall'iterazione di input. Cancellate il programma in memoria. Passiamo ad un'altra funzione delle variabili a stringa: CHR$.
    CHR$ è una funzione che converte un numero in un carattere sulla tastiera. La maggior parte dei caratteri sulla tastiera corrisponde a un numero. Immettete il programma seguente:

```
100 INPUT N
110 PRINT CHR$(N)
120 GOTO 100
130 END
```

    Eseguite ora il programma e alla richiesta di immissione di dati, battete 65. Che cosa è successo?

    ....

    Il programma continuerà a ripetere l'iterazione per tutto il tempo che vogliamo. Questa volta proviamo 66. Che cosa è successo?

    ....

    Provate anche 49, 50 e 255.

17. Giocate un po' con questo programma provando ad immettere vari numeri. Mantenete i numeri nei limiti da 32 a 255. Molto presto dovreste accorgervi che potete riferirvi ad un carattere sia tramite il carattere stesso, che tramite un numero di posizione nel gruppo di caratteri. Interrompete il programma.

18. Per vedere una parte del gruppo di caratteri ASCII, cancellate la memoria e battete il programma seguente. I simboli che appariranno sullo schermo sono alcuni dei caratteri grafici del C-64.

```
100 FOR N=65 TO 122
110 PRINT CHR$(N);" ";
120 NEXT N
130 END
```

Potete vedere l'intero gruppo di caratteri ASCII consultando l'appendice C della *Guida di riferimento per il programmatore*. Dopo aver eseguito il programma, premendo il tasto C= (in basso a sinistra sulla tastiera) e SHIFT si cambiano i caratteri sullo schermo. Ripetendo l'operazione, lo schermo cambia di nuovo. Si possono così vedere facilmente le differenze tra modo maiuscolo/minuscolo e maiuscolo/grafico.

19. Passiamo ora ad un altro argomento. Prima cancellate il programma in memoria. ASC è la funzione che converte un carattere nell'equivalente codice di carattere usato dal computer.
    Battete il programma seguente:

```
100 INPUT A$
110 LET X=ASC(A$)
120 PRINT X
130 GOTO 100
140 END
```

Eseguite il programma e alla richiesta di input, battete Z. Che cosa è successo?

....

Questa nuova funzione ASC(A$) è esattamente il contrario di CHR$(N). Usate varie lettere e numeri e confrontate i risultati con quelli che avete ottenuto al punto 16.

20. C'è ancora un dettaglio da definire. Il computer si dovrebbe ancora trovare in attesa di un'immissione di A$. Questa volta battete PATATE. Annotate quello che viene scritto in risposta. Ora provate PISELLI. Che cosa è successo?

....

L'unica somiglianza fra le due parole è che entrambe hanno la stessa lettera iniziale. Torneremo più tardi su questo concetto.

21. I numeri possono essere considerati stringhe di cifre. Così un nume-

ro può essere immesso come una stringa di cifre. VAL è la funzione che converte queste stringhe di cifre nel loro valore numerico. Cancellate il programma in memoria e battete:

```
100 INPUT A$,B$
110 PRINT A$ + B$
120 PRINT VAL(A$) + VAL(B$)
130 END
```

Visualizzate il programma ed eseguitelo. Alla richiesta di input battete 5,6. L'uscita dalle righe 110 o 120 è la somma aritmetica di 5 e 6?

....

Notate che uno spazio in più prima dell'11 è riservato al segno −.

22. La funzione STR$ converte un numero nella sua stringa di cifre. Cancellate il programma in memoria e battete:

```
100 INPUT A
110 LET A$=STR$(A)
120 PRINT MID$(A$,1,4)
130 END
```

Visualizzate il programma ed eseguitelo. Alla richiesta di input, battete 1234. Sono state visualizzate tutte e quattro le cifre battute?

....

Anche qui il primo spazio è riservato al segno −. La stringa che rappresenta il numero 1234 è perciò lunga 5 caratteri.

23. Nel capitolo precedente abbiamo usato array numerici. Nei programmi BASIC potete usare anche array stringa. Battete

```
100 DIM A$(6)
110 FOR I=1 TO 6
120 READ A$(I)
130 PRINT A$(I);
140 NEXT I
150 DATA 8,LEO,9,ANNA,10,PIPPO
160 END
```

Eseguite il programma. Scrivete cosa è uscito.

....

Qui 8,9 e 10 sono stringhe di caratteri in A$, un array di stringa. Per sommare 8,9 e 10 aggiungere la riga 145:

```
145 PRINT VAL(A$(1))+VAL(A$(3))+VAL(A$(5))
```

Eseguite il programma. La somma è corretta?

....

Cambiate le righe 100, 120 e 130:

```
100 DIM A(6)
120 READ A(I)
130 PRINT A(I);
```

Visualizzate il programma ed eseguitelo. Che numero appare?

....

Che messaggio d'errore viene fuori?

....

24. Con ciò si conclude il lavoro sul computer per questo capitolo. Spegnete il computer e la TV e passate alla sezione seguente.

8.3 Analisi

INPUT E OUTPUT DI STRINGHE

Come avete già visto, un gruppo di caratteri racchiuso tra virgolette è chiamato stringa. Le virgolette non fanno però parte della stringa. L'idea nuova di questo capitolo è che la stringa può essere trattata come una variabile: la variabile a stringa.
La variabile a stringa si identifica aggiungendo un segno dollaro ($) al nome: CORTILE$, CASSA$ e C$ sono nomi di variabili a stringa.
L'ingresso e l'uscita delle variabili a stringa sono trattati allo stesso modo di quelli delle variabili numeriche. Nelle stesse istruzioni BASIC pos-

siamo mescolare variabili numeriche e a stringa. Esempi ne possono essere

```
100 PRINT A$,X,Y,Z$
110 INPUT M$,N
120 READ A$,B$,Z
```

Dovete fare attenzione che l'immissione di dati sia nelle istruzioni INPUT che in quelle READ concordi con il tipo di variabile dato. Nella riga 110 qui sopra, il computer si aspetterà una stringa di caratteri e un numero. Dovete però rendervi conto che potete battere 123456789 e se il computer si aspetta una stringa identificherà questa quantità come una stringa, e non come un numero. Il motivo è che la stringa, come è stato fatto notare, è formata da caratteri, e anche i simboli da 0 a 9 fanno parte del gruppo standard di caratteri che sarà trattato più avanti. Se, d'altra parte, il computer si aspetta un numero e voi battete ABCDEFGHI, otterrete un errore.
Anche gli array stringa si possono dimensionare. Ciascun elemento di un array stringa è una stringa di 255 caratteri al massimo. Per esempio, se eseguite il seguente programma:

```
100 DIM A$(60)
110 LET A$(51)="12"
120 LET A$(52)="34"
130 PRINT A$(51) + A$(52)
140 PRINT VAL(A$(51)) + VAL(A$(52))
150 END
```

la riga 130 stamperà 1234 poiché il segno + unisce le due stringhe A$(51) e A$(52) assieme. La riga 140 stamperà 46, la somma di 12 e 34. Questo esempio puntualizza la differenza tra stringa e variabili numeriche e tra stringa e array numerici.

FUNZIONI SU STRINGHE

La funzione LEN viene usata per determinare la lunghezza di una stringa. Se, per esempio, A$ = "TANTO VA LA GATTA", allora LEN(A$) = 17. Si noti che gli spazi contano come caratteri. Si può anche avere una stringa "nulla". Se A$ = "" (non c'è nulla fra le virgolette), allora LEN(A$) = 0.

LEN(A$) dà il numero di caratteri in A$

Una sottostringa è un pezzo o segmento di stringa. Vi sono vari modi di lavorare con le sottostringhe. Considerate il programma seguente:

```
100 LET A$="GIANNI A. BIANCHI"
110 PRINT MID$(A$,8,6)
120 END
```

L'espressione MID$(A$,8,6) identifica la sottostringa di A$ formata da sei caratteri a partire dall'ottavo carattere. Se il programma fosse avviato, verrebbero stampati i caratteri A. BIA.

MID(A$,I,J) dà J caratteri della stringa A$ a partire dal carattere I

Le variabili a stringa possono essere confrontate in istruzioni IF THEN. Il confronto è fatto tramite l'ordinamento alfabetico. Così A<B perché A viene prima di B nell'alfabeto. CANE<GATTO, CASA>AUTOMOBILE, BURRO<BURRONE, e così via.
Per capire le funzioni stringa, dovete conoscere il gruppo standard di caratteri ASCII (l'abbreviazione di *American Standard Code for Information Interchange*, il codice standard americano per lo scambio di informazioni). Questo gruppo di caratteri è usato dalla maggior parte dei computer ed è formato da centoventotto caratteri numerati da 0 a 127. Per la lista completa dei caratteri ASCII fate riferimento al manuale di consultazione del Commodore 64.
Nel modo maiuscolo/grafico le lettere maiuscole da A a Z sono numerate da 65 a 90. Nel modo maiuscolo/minuscolo le lettere minuscole da a a z corrispondono ai numeri da 65 a 90. I numerali da 0 a 9 sono numerati da 48 a 57. Gli altri caratteri numerati comprendono la punteggiatura, gli operatori aritmetici (+, −, *, ecc.) e altri caratteri speciali.
Poiché ci sono due modi di scrittura, ciascun numero può corrispondere a due simboli diversi.
Due funzioni a stringa operano con il gruppo di caratteri ASCII. La prima CHR$(N) riporta l'N-esimo carattere del gruppo di caratteri ASCII. Per esempio CHR$(65) = A, CHR$(90) = Z e via di seguito. Possiamo anche capovolgere le cose e convertire un carattere nel suo numero ASCII. Questo si ottiene con la funzione ASC(A$). Per esempio, ASC("A") = 65 e ASC("Z") = 90.
Supponiamo A$ = "AEROPLANO". Che cosa sarà ASC(A$)? Siccome la lunghezza della stringa è maggiore di uno, viene considerato solo il primo carattere. In questo caso il primo carattere è A e ASC(A$) = 65.
Si usano due funzioni per trattare con numeri come stringhe di cifre. STR$(−945) dà come risultato una stringa i cui caratteri sono il segno −, 9, 4, 5. La funzione VAL invece dà come risultato il valore numerico di una stringa i cui caratteri sono cifre. Così VAL("112233445566") dà 1.12233446E+11.

8.4 Esempi di programmi

ESEMPIO 1 — INVERSIONE DI STRINGHE

Il compito è quello di scrivere un programma che richieda l'immissione di una stringa e poi la emetta in ordine inverso. Per prima cosa dobbiamo predisporre l'ingresso della stringa.

```
100 INPUT A$
```

Le poche righe che seguono faranno scrivere la stringa in ordine inverso.

```
110 FOR X=LEN(A$) TO 1 STEP -1
120 PRINT MID$(A$,X,1,);
130 NEXT X
```

L'iterazione cammina all'indietro, dalla lunghezza della stringa fino ad 1. La funzione MID$(A$,X,1) identifica in A$ la sottostringa formata da 1 carattere che inizia al carattere numero X. Con ciò viene isolato un unico carattere.
Con un'istruzione END aggiunta, il programma completo è

```
100 INPUT A$
110 FOR X=LEN(A$) TO 1 STEP -1
120 PRINT MID$(A$,X,1);
130 NEXT X
140 END
```

Provate ad eseguire questo programma usando il vostro nome in ingresso.

ESEMPIO 2 — CONTEGGIO DI PAROLE

Il numero di parole in una frase può essere determinato dal numero di spazi (presumendo che l'unico scopo dello spazio sia quello di separare le parole). Il programma che segue stampa il numero di parole contenute nella stringa in ingresso.

```
100 INPUT A$
110 LET S=0
120 FOR I=1 TO LEN(A$)
130 IF MID$(A$,I,1)<>" " THEN 150
140 LET S=S+1
150 NEXT I
160 PRINT "NUMERO DELLE PAROLE = ";S+1
170 END
```

Studiate il programma fino a che non vedete esattamente come funziona. Provatelo battendo una frase. Verificate che funzioni in modo corretto.

ESEMPIO 3 — CODIFICA DI FRASI

Supponiamo di volere un programma che codifichi una frase. Un modo facile di costruire un codice (che tra parentesi potrebbe essere rapidamente scoperto con i computer) è di sostituire ciascun carattere del messaggio con un altro carattere. Questo si ottiene in modo estremamente facile facendo riferimento al gruppo di caratteri ASCII. Tuttavia, facciamolo "partendo da zero".
La prima parte del programma richiede la stringa da codificare e stabilisce lo schema di conversione.

```
100 LET B$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ ."
110 LET C$ = "ETAVZBHCW KPSYDFGXIMJLONQU.R"
120 INPUT A$
```

B$ contiene i caratteri che possono essere usati nella stringa in ingresso da codificare. C$ è la chiave di sostituzione. Una A nella stringa immessa viene sostituita da una E, F è sostituita da B, J da uno spazio, e via discorrendo.
Possiamo ora esaminare ciascun carattere e operare le sostituzioni.

```
130 FOR I=1 TO LEN(A$)
140 FOR J=1 TO 29
150 IF MID$(A$,I,1) <> MID$(B$,J,1) THEN 180
160 PRINT MID$(C$,J,1);
170 GOTO 190
180 NEXT J
190 NEXT I
```

L'iterazione esterna I fa passare ogni carattere di A$. L'iterazione interna J confronta l'I-esimo carattere di A$ con il carattere in B$ fino a che si trova una corrispondenza al J-esimo carattere. Quando ciò avviene, viene stampato il J-esimo carattere codificato di C$ e il programma va avanti al successivo carattere di A$.
Terminiamo il programma con

```
200 PRINT
210 END
```

Il programma completo è

```
100 LET B$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ ."
110 LET C$ = "ETAVZBHCW KPSYDFGXIMJLONQU.R"
120 INPUT A$
130 FOR I=1 TO LEN(A$)
140 FOR J=1 TO 29
150 IF MID$(A$,I,1) <> MID$(B$,J,1) THEN 180
160 PRINT MID$(C$,J,1);
170 GOTO 190
180 NEXT J
190 NEXT I
200 PRINT
210 END
```

Il codice può essere cambiato disponendo in un altro modo i caratteri in C$. Può essere interessante provocare il programma e vedere come si presenta un messaggio codificato.

ESEMPIO 4 — TABELLA DELLE VENDITE

La seguente è una tabella delle vendite al dettaglio di un negozio di computer. Le vendite sono fornite da ciascun venditore ogni trimestre.

	Trimestre			
	1	2	3	4
Aldo	7	9	11	17
Marco	5	10	15	20
Pippo	8	7	21	14

Questa tabella può essere rappresentata come un array stringa di tre righe e 5 colonne. Il vostro compito è scrivere un programma che cerchi il nome del venditore, calcoli il totale delle vendite di questo venditore e stampi il risultato.
Prima dimensionate l'array:

```
100 DIM UNIVEN$(3,5)
```

Poi inserite i dati nel programma

```
110 DATA ALDO,7,9,11,17
120 DATA MARCO,5,10,15,20
130 DATA PIPPO,8,7,21,14
```

Ora immettete i dati nell'array stringa UNIVEN$

```
140 FOR R=1 TO 3
150 FOR C=1 TO 5
160 READ UNIVEN$(R,C)
170 NEXT C
180 NEXT R
```

Ora chiedete il nome del venditore

```
190 PRINT "IMMETTERE IL NOME DEL VENDITORE ";
200 INPUT N$
```

Successivamente determinate il numero di riga del venditore

```
210 FOR R=1 TO 3
220 IF N$=UNIVEN$(R,1) THEN  (Somma le vendite)
230 NEXT R
```

Se un nome è stato inserito erroneamente, avrete bisogno delle seguenti righe:

```
240 PRINT N$;" NON E' UN VENDITORE "
250 GOTO 190
```

Ora calcolate il numero totale dei computer venduti

```
300 REM CALCOLA LE VENDITE
310 LET S=0
320 FOR C=2 TO 5
330 LET S=S+VAL(UNIVEN$(R,C))
340 NEXT C
350 PRINT N$;" HA VENDUTO ";S;" COMPUTER"
360 END
```

Poiché gli array che state usando sono array stringa, i numeri nell'array sono trattati come stringhe. Per convertirli dovete usare la funzione VAL nella riga 330. Una volta che avete la tabella come array stringa, potete rispondere a molti tipi di domanda programmando in BASIC. La tabella può, naturalmente, essere più grande.
Segue il programma completo.

```
100 DIM UNIVEN$(3,5)
110 DATA ALDO,7,9,11,17
```

```
120 DATA MARCO,5,10,15,20
130 DATA PIPPO,8,7,21,14
140 FOR R=1 TO 3
150 FOR C=1 TO 5
160 READ UNIVEN$(R,C)
170 NEXT C
180 NEXT R
190 PRINT "IMMETTERE IL NOME DEL VENDITORE ";
200 INPUT N$
210 FOR R=1 TO 3
220 IF N$=UNIVEN$(R,1) THEN 300
230 NEXT R
240 PRINT N$;" NON E' UN VENDITORE "
250 GOTO 190
300 REM CALCOLA LE VENDITE
310 LET S=0
320 FOR C=2 TO 5
330 LET S=S+VAL(UNIVEN$(R,C))
340 NEXT C
350 PRINT N$;" HA VENDUTO ";S;" COMPUTER"
360 END
```

8.5 Problemi

1. Scrivere un programma che richieda l'immissione di una stringa. Quindi scriva la stringa in una colonna verticale di caratteri.

2. Scrivere un programma che richieda l'immissione di una stringa e che scriva poi la stringa in diagonale in modo che ciascun carattere si trovi una riga sotto e un carattere a destra del precedente.

3. Scrivere un programma per contare il numero di vocali presenti in una stringa immessa.

4. Scrivere un programma che richieda l'immissione di una stringa e che poi stampi le parole della stringa in una colonna verticale.

5. Chiedere l'immissione di una frase. Generare da questa frase una nuova stringa dalla quale siano stati tolti tutti gli spazi. Poi stampare la nuova stringa.

6. Scrivere un programma che richieda l'immissione di una stringa formata da lettere maiuscole. Quindi converta le lettere maiuscole in minuscole e stampi la stringa. Potete aver bisogno di far riferimento alla tabella dei caratteri ASCII nell'appendice C del vostro manuale di consultazione.

7. Si presuma che debbano essere battute cinque frasi. Scrivere un programma per contare il numero di volte che la parola IL compare nelle cinque frasi.

8. Se al segnale di ingresso del programma qui sotto esposto viene battuta la stringa ABCDEFGH quale sarà l'uscita?

```
100 INPUT A$
110 FOR J=1 TO LEN(A$) STEP 2
120 PRINT MID$(A$,J,1);
130 NEXT J
140 END
```

9. Scrivere un programma che richieda l'immissione di una stringa e conti il numero di volte che il carattere I è seguito dal carattere N.

10. Vogliamo contare la frequenza dell'apparizione di ciascuna delle 26 lettere dell'alfabeto (si può presumere che siano tutte maiuscole) in dieci frasi che devono essere immesse da tastiera. Non contate gli spazi o i segni di interpunzione. Scrivete un programma che calcoli e scriva una tabella formata da ciascuna delle lettere e dal numero di volte che queste appaiono nelle frasi.

11. Modificate il programma dell'esercizio 4 per determinare le vendite di ciascun venditore senza cercarne il nome.

12. Modificate il programma dell'esercizio 4 per determinare il totale delle vendite nel terzo trimestre.

8.6 Test di apprendimento

1. Come sono identificate in BASIC le variabili a stringa?

   ....

2. Se A$ = "CANE" e B$ = "CANESTRO" allora A$>B$. È vero o falso?

   ....

3. Se A$ = "TANTO VA LA GATTA AL LARDO" scrivere una funzione che estragga LA GATTA.

   ....

4. Scrivere un programma che richieda l'immissione di una stringa e poi continui a stampare in risposta la stringa con un carattere cancellato ogni volta fino a quando non sia rimasto più niente. Se, per esempio, si batte FETTA DI TORTA, il computer deve rispondere con

```
FETTA DI TORTA
FETTA DI TORT
FETTA DI TOR
FETTA DI TO
FETTA DI T
FETTA DI
FETTA DI
FETTA D
FETTA
FETTA
FETT
FET
FE
F
```

5. Che cosa verrà stampato se verrà eseguito il seguente programma?

```
100 FOR N=65 TO 90
110 FOR M=65 TO N
120 PRINT CHR$(M);
130 NEXT M
140 PRINT
150 NEXT N
160 END
```

Capitolo 9

Funzioni e subroutine

9.1 Obiettivi

In questo capitolo impareremo come il computer possa essere programmato per eseguire delle sotto-operazioni. Ciò può essere fatto sia tramite segmenti di programma che attraverso delle speciali istruzioni.

FUNZIONI

Abbiamo visto in precedenza delle funzioni incorporate nel BASIC. Ora impareremo come definire funzioni nostre per poter svolgere qualunque compito particolare.

SUBROUTINE

I sottoprogrammi (subroutine) sono molto utili quando devono essere ripetute delle operazioni complicate. Vedremo come si possono costituire delle subroutine e usarle in programmi BASIC.

ESEMPI DI PROGRAMMI

Come il solito, applicherete ciò che avete imparato in alcuni esempi di programma.

9.2 Esercizi di scoperta

1. Accendete il computer e la TV. Immettete il seguente programma:

```
100 DEF FNA(X)=5*X+4
110 LET X=2
120 LET Y=5*X+4
130 PRINT Y,FNA(2)
140 END
```

Eseguite il programma e registratene qui sotto l'uscita.

....

2. Cambiate la riga 130 come segue:

```
130 PRINT Y,FNA(X)
```

Visualizzate il programma in memoria. Che cosa pensate che succeda se si esegue questo programma?

....

Eseguite il programma. Che cosa è successo?

....

3. Battete

```
110 LET X=5
```

Visualizzate il programma e studiatelo. Ora quale sarà l'uscita se si esegue il programma?

....

Controllate se avevate ragione. Eseguite il programma e annotate quello che è successo.

....

4. Cambiate la riga 130 come segue:

```
130 PRINT Y,FNA(5)
```

Visualizzate il programma. Che cosa pensate che faccia ora il programma?

....

Eseguite il programma e annotatene l'uscita.

....

5. Notate che le espressioni dopo il segno di uguale nelle righe 100 e 120 del vostro programma sono le stesse. In una delle versioni del programma abbiamo stampato Y e FNA(X) e abbiamo visto che erano uguali. Esaminiamo un po' questa informazione. Cancellate il programma in memoria. Quindi immettete il programma seguente.

```
100 DEF FNA(X)=X↑2
110 DEF FNB(X)=3*X
120 DEF FNC(X)=X+2
130 LET X=1
140 PRINT FNA(X);FNB(X);FNC(X)
150 END
```

Studiate attentamente il programma. Che cosa pensate che sarà stampato se il programma verrà eseguito?

....

Ora eseguite il programma e scrivete quello che è successo.

....

Senza cambiare il programma, che cosa pensate che succederebbe se si sostituisse X con 1 nelle espressioni nella parte destra delle righe 100, 110 e 120 e quindi si eseguisse il programma? Scrivete i numeri che se ne otterrebbero.

....

6. Battete

```
130 LET X=2
```

Visualizzate il programma. Che cosa verrà stampato se il programma venisse eseguito?

....

Controllate se eravate nel giusto. Eseguite il programma e annotatene qui sotto i risultati.

....

7. Battere

```
130 LET X=3
```

Ora che cosa succederà se il programma verrà avviato?

....

Verificate la vostra risposta eseguendo il programma e annotando quello che è successo.

....

8. Adesso passiamo a esplorare qualche nuova idea con questo programma. Battete

```
130 LET X=1
140 PRINT FNC(X+4);FNB(2);FNA(X)
```

Visualizzate il programma. Scrivete quello che pensate che verrà stampato se il programma sarà eseguito.

....

Avviate il programma e annotatene l'uscita.

....

9. Proviamo una variazione leggermente diversa sul tema che siamo venuti esplorando finora. Battete

```
140 PRINT FNA(X);FNB(FNA(X))
```

Visualizzate il programma e studiatelo attentamente. Provate ad immaginare quello che verrà scritto quando il programma sarà eseguito. Scrivete qui sotto la vostra risposta.

....

Eseguite il programma e vedete un po' se avevate ragione. Scrivete quello che è successo.

....

10. Un'ultima cosa su questo argomento. Battete

```
130 LET X=4
140 PRINT FNA(X);FNC(X);FNA(SQR(X))
```

Ora che cosa succederà nel programma?

....

Eseguite il programma e registrate quello che è successo.

....

11. Cancellate il programma in memoria. Immettete il programma seguente:

```
100 PRINT "A";
110 GOSUB 200
120 PRINT "B";
130 GOSUB 300
140 PRINT "C";
150 END
200 PRINT 1;
210 RETURN
300 PRINT 2;
310 RETURN
```

Questo programma ha due nuove istruzioni che non avete ancora visto finora. Sono GOSUB e RETURN. Il programma in sé ha la sola funzione di farvi fare un po' di pratica con queste nuove istruzioni. Eseguite il programma e registratene l'uscita.

....

Confrontate quello che è stato stampato con le righe che hanno provocato tale uscita.

....

12. A quale riga l'istruzione GOSUB della riga 110 trasferisce il programma? (Suggerimento: osservate l'uscita al punto 11).

....

13. A quale istruzione l'istruzione RETURN della riga 210 trasferisce il programma? (Suggerimento: esaminate ancora l'uscita al punto 11).

....

14. I numeri di riga qui sotto indicano il flusso del programma mentre viene eseguito.

Numero di riga	Che cosa succede
100	Stampa A
110	Trasferisce alla riga 200
200	Stampa 1
210	Trasferisce alla riga 120
120	Stampa B
130	Trasferisce alla riga 300
300	Stampa 2
310	Trasferisce alla riga 140
140	Stampa C
150	Fine del programma

Studiate attentamente questo schema e seguitelo nel programma. Riuscite già a capire lo scopo delle istruzioni GOSUB e RETURN? Che ne pensate del posizionamento dell'istruzione END?

....

15. Cancellate il programma in memoria e immettete il seguente :

```
100 REM PROGRAMMA DIMOSTRATIVO DELL'USO DI SUBROUTINE
110 DIM X(4)
120 FOR I=1 TO 4
130 READ X(I)
140 NEXT I
150 REM ORDINA
160 GOSUB 300
170 REM VISUALIZZA L'ARRAY ORDINATO
180 FOR I=1 TO 4
190 PRINT X(I);
200 NEXT I
210 PRINT
```

```
220 LET X(3)=7
230 REM ORDINA DI NUOVO
240 GOSUB 300
250 REM VISUALIZZA L'ARRAY ORDINATO
260 FOR I=1 TO 4
270 PRINT X(I);
280 NEXT I
290 END
295 DATA 2,1,5,6
300 REM SUBROUTINE CHE ORDINA L'ARRAY
310 FOR I=1 TO 3
320 IF X(I+1) > X(I) THEN 370
330 LET TEMP=X(I+1)
340 LET X(I+1)=X(I)
350 LET X(I)=TEMP
360 GOTO 310
370 NEXT I
380 RETURN
```

Visualizzate il programma e controllate di averlo immesso in modo esatto. Questo programma fornisce un esempio di come possa essere usata una subroutine. La subroutine che si trova nelle righe da 300 a 380 mette in ordine l'array X in modo ascendente. Eseguite il programma e annotatene l'uscita.

....

Notate che l'array originale è

```
2 1 5 6
```

Potete vedere questo controllando l'istruzione DATA nella riga 295. Nella riga 160 il programma salta alla subroutine e viene effettuato un ordinamento dei numeri. Dopo che il programma è tornato alla riga 170, l'array ordinato è

```
1 2 5 6
```

Nella riga 220 cambiamo il terzo elemento dell'array, quindi si salta alla subroutine per un altro ordinamento. Dopo che si è tornati alla riga 250, viene stampato l'array ordinato

```
1 2 6 7
```

Infine il comando END della riga 290 fa sì che il programma fermi l'esecuzione. È chiaro che potremmo mettere in ordine l'array X tante volte quante vogliamo, semplicemente inserendo l'istruzione GOSUB 300. È certamente un sistema più efficiente che scrivere le istruzioni per l'ordinamento tutte le volte che ne abbiamo bisogno.

16. Vediamo ora un'altra istruzione. Cancellate la memoria e battete il seguente programma.

```
100 PRINT "IMMETTERE UN NUMERO TRA 1 E 5 "
110 PRINT "BATTERE 5 PER FINIRE "
120 INPUT N
130 ON N GOSUB 1000,2000,3000,4000
140 IF N=5 THEN END
150 PRINT
160 GOTO 100
1000 PRINT "UNO"
1010 RETURN
2000 PRINT "DUE"
2010 RETURN
3000 PRINT "TRE"
3010 RETURN
4000 PRINT "QUATTRO"
4010 RETURN
```

Quale parola verrà visualizzata se si immette 3?

....

Eseguite il programma. Avevate ragione?

....

17. E con questo è terminato il lavoro sul computer per questo capitolo. Spegnete il computer e la TV.

9.3 Analisi

FUNZIONI

L'istruzione DEF (abbreviazione di "definisci") ci permette di avere nei programmi BASIC delle funzioni definite dall'utente in aggiunta a quelle

(come SQR, INT, ecc.) già incorporate nel linguaggio. La forma di tutte le istruzioni DEF è la stessa.

N° riga DEF FN *variabile (argomento) = espressione*

```
100 DEF FNP(X)=X↑2 - 3*X
```

Siccome nomi diversi di variabili potrebbero seguire FN, potremmo avere diverse funzioni definite in un unico programma.
Se in un programma fosse usata l'istruzione DEF della riga 100, e più tardi fosse usata l'espressione FNP(2), il computer identificherebbe che FNP è stata definita alla riga 100 e sostituirebbe con 2 la X che si trova alla destra del segno di uguale nell'istruzione DEF, col risultato che

FNP(2) = −2

Parimenti, se VOLTE = 5, allora

FNP(VOLTE) = 10

Nelle istruzioni DEF possono essere usate le funzioni incorporate del BASIC. Per esempio,

```
100 DEF FNB(Y)=SQR(Y↑1.5) + 3*Y
```

va bene. Inoltre in un'istruzione DEF potete usare altre funzioni definite. Per esempio,

```
100 DEF FNB(Y)=FNA(Y) + SQR(Y)
```

è esatta ammesso che la funzione FNA sia stata in precedenza definita. Lo scopo principale delle funzioni definite dall'utente è di semplificare la programmazione evitando l'uso ripetuto di espressioni complicate. Il programmatore attento deve essere consapevole delle opportunità di risparmiare sforzi mediante l'uso delle istruzioni DEF.

Definite le vostre funzioni con un'istruzione DEF

SUBROUTINE

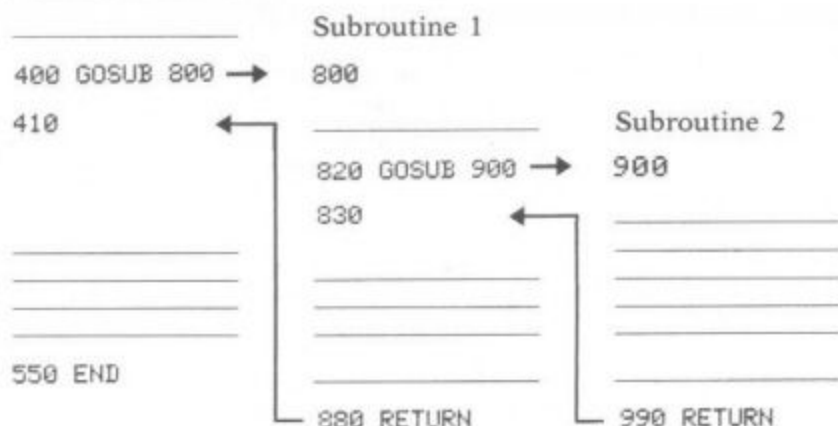
Una limitazione delle istruzioni DEF è che non possono essere più lunghe di una riga. In un programma possono sorgere però delle situazioni più complicate nelle quali vogliamo eseguire lo stesso procedimento molte

volte. È qui che le subroutine tornano utili. Il diagramma seguente indica come una subroutine possa essere usata in un programma.

Il programma principale comincia	________
	________
	________
	200 GOSUB 1000
	210 ________
	________
	________
	350 GOSUB 1000
	360 ________
Il programma principale termina	430 END
Inizia la subroutine	1000 REM SUBROUTINE
	________
	________
La subroutine termina	1150 RETURN

Se il programma – tipo suddetto venisse eseguito, quando il computer raggiunge il GOSUB nella riga 200, salterebbe all'inizio della subroutine alla riga 1000. La subroutine verrebbe eseguita e, trovato il RETURN alla riga 1150, il controllo passerebbe alla riga con il numero immediatamente superiore dopo il GOSUB che ci ha messi nella subroutine. In questo caso il programma salterebbe indietro alla riga 210. Poi il computer andrebbe avanti per il programma principale fino al GOSUB della riga 350, che farebbe di nuovo passare il controllo alla subroutine della riga 1000. Questa volta il RETURN farebbe saltare indietro alla riga 360 del programma.
Naturalmente, potremmo usare GOSUB 1000 tutte le volte che si vuole nel programma, o potremmo avere tutte le subroutine che ci servono. Generalmente la parte superiore del programma è il programma principale e le subroutine sono raggruppate assieme alla fine. C'è un buon motivo per fare questo. Vogliamo eseguire le subroutine soltanto quando queste sono richiamate da un GOSUB. Così, dopo che il programma principale è finito, mettiamo un'istruzione END nel programma.

Saltare alle subroutine con GOSUB

È possibile, e talvolta desiderabile, saltare ad una subroutine da un'altra subroutine. Il diagramma qui sotto indica come il computer tratta un tale evento.

Si noti che il controllo passa da 400 a 800, giù fino a 820, a 900, e giù fino al RETURN della riga 990. Naturalmente, il problema qui è: il RETURN ci farà tornare alla riga 410 o alla riga 830? La regola è che il RETURN ci riporta all'istruzione successiva a quella in cui si trova il GOSUB che ci ha messi nella subroutine. Siamo nella subroutine 2 a causa del GOSUB della riga 820; di conseguenza il RETURN della riga 990 ci farà tornare alla riga 830. La stessa regola si applica quando raggiungiamo il RETURN alla riga 880. A quel punto ci troviamo nella subroutine 1 e ci siamo arrivati dal GOSUB della riga 400. Così, il RETURN della riga 880 ci riporta indietro alla riga 410.

Ritornare indietro dalle subroutine con RETURN

L'istruzione ON...GOSUB permette di richiamare varie subroutine sulla base del valore numerico di una variabile. Per esempio l'istruzione

```
140 ON N GOSUB 300,400,500
```

manderà alla subroutine che si trova in 300 se N è 1, alla subroutine in 400 se N è 2, o a quella in 500 se N è 3.
A questo punto può non esservi ancora chiaro perché le subroutine siano importanti. La necessità di usarle diventa più evidente a mano a mano che si acquista più esperienza di programmazione. È sufficiente ora mettere in luce il fatto che le subroutine sono estremamente importanti e sono considerate come uno degli strumenti più potenti a disposizione del programmatore.

9.4 Esempi di programmi

ESEMPIO 1 — ARROTONDAMENTO AI CENTESIMI

Molte applicazioni commerciali che trattano valori in dollari implicano la stampa dei risultati dei calcoli in dollari e centesimi. Siccome il computer normalmente elabora sette cifre significative, potremmo ottenere in uscita un importo come 23.15793. Questo ha un aspetto strano e per risolvere il problema dovremmo arrotondare la cifra al centesimo più prossimo, cioè 23.16.
Questa è un'applicazione ideale per una funzione definita dall'utente. Scriviamo un programma che produca, quando eseguito, l'uscita seguente:

PREZZO DI LISTINO? (Si immette il prezzo)
SCONTATO DEL 10% È (Il computer stampa il prezzo scontato)
SCONTATO DEL 15% È (Il computer stampa il prezzo scontato)
SCONTATO DEL 20% È (Il computer stampa il prezzo scontato)

Tutti i valori in dollari scritti dal computer devono essere arrotondati al centesimo più vicino.
Prima dobbiamo definire una funzione che effettui l'arrotondamento. Una tale funzione è

```
100 DEF FNR(X)=INT(X*100+.5)/100
```

Per vedere come questa regola funzioni, supponiamo X = 23.1597. Possiamo seguire questo valore attraverso l'espressione per vedere quello che avviene.

$$X*100 = 2315.97$$
$$X*100+0.5 = 2316.47$$
$$INT(X*100+0.5) = 2316$$
$$INT(X*100+0.5)/100 = 23.16$$

Perciò 23.1597 è stato giustamente arrotondato per eccesso a 23.16.
Come secondo esempio, supponiamo X = 23.1547. Allora

$$X*100 = 2315.47$$
$$X*100+0.5 = 2315.97$$
$$INT(X*100+0.5) = 2315$$
$$INT(X*100+0.5)/100 = 23.15$$

con il risultato che 23.1547 è stato correttamente arrotondato per difetto a 23.15.
Le successive poche righe di programma si spiegano da sole.

```
110 PRINT "PREZZO DI LISTINO";
120 INPUT Z
130 PRINT "SCONTATO DEL 10% E'";FNR(.90*Z)
140 PRINT "SCONTATO DEL 15% E'";FNR(.85*Z)
150 PRINT "SCONTATO DEL 20% E'";FNR(.80*Z)
```

Se si desidera, possiamo tornare con un'iterazione all'inizio con

```
160 GOTO 110
```

e poi terminare il programma

```
170 END
```

Il programma completo è

```
100 DEF FNR(X)=INT(X*100+.5)/100
110 PRINT "PREZZO DI LISTINO";
120 INPUT Z
130 PRINT "SCONTATO DEL 10% E'";FNR(.90*Z)
140 PRINT "SCONTATO DEL 15% E'";FNR(.85*Z)
150 PRINT "SCONTATO DEL 20% E'";FNR(.80*Z)
160 GOTO 110
170 END
```

Nelle righe 130, 140 e 150 viene usata la funzione definita. Con uno sconto del 10 per cento, il prezzo di vendita è il 90 per cento del prezzo originale Z. Perciò stampiamo FNR(0.9 * Z), che arrotonda il valore al centesimo più prossimo come desiderato. Notate l'economia che si realizza usando la funzione definita, piuttosto che scrivere l'espressione della riga 100 tutte le volte che vogliamo far stampare un importo arrotondato in dollari e centesimi.

ESEMPIO 2 — MOQUETTE

Vogliamo scrivere un programma che usi una subroutine per calcolare il prezzo d'acquisto della moquette. Supponiamo che ci siano quattro tipi di moquette e ognuno di questi venga scontato a seconda della quantità di moquette ordinata. Presumiamo che la tabella dei prezzi sia la seguente:

Tipo	**1**	**2**	**3**
A	L 10000	L 8500	L 7250
B	L 13250	L 12000	L 9750
C	L 16000	L 14000	L 11250
D	L 20000	L 17200	L 15250

Le colonne si riferiscono a:

1. I primi 15 metri quadri
2. La parte dell'ordine che eccede 15 metri quadri, ma che non supera i 25
3. Tutto quello che supera i 25 metri quadri.

Quando viene eseguito, il programma deve produrre il seguente output:

QUANTE STANZE? (Si immette)
PER OGNI STANZA, IMMETTERE LUNGHEZZA
E LARGHEZZA IN CENTIMETRI
SEPARATE DA UNA VIRGOLA

STANZA	DIMENSIONI
1	(Si immettono)
2	(Si immettono)

(L'iterazione prosegue fino a che non si siano immessi i dati di tutte le stanze)

(numero di) METRI QUADRI RICHIESTI

TIPO DI MOQUETTE	COSTO
A	(Il computer stampa, ecc.)
B	
C	
D	

Prima di iniziare la stesura del programma, dovremmo pensare a cosa vogliamo ottenere. Possiamo ad esempio usare una funzione definita dall'utente per effettuare un arrotondamento alle mille lire. Cominciamo così il programma con quella funzione.

```
100 DEF FNR(X)=INT(X/1000)*1000
```

Le righe successive seguono senza difficoltà.

```
110 PRINT "QUANTE STANZE";
120 INPUT N
130 PRINT "PER OGNI STANZA, IMMETTERE LUNGHEZZA"
140 PRINT "E LARGHEZZA IN CENTIMETRI"
150 PRINT "SEPARATE DA UNA VIRGOLA"
160 PRINT
170 PRINT "STANZA","DIMENSIONI"
180 PRINT
```

Ora siamo pronti per chiedere l'immissione delle dimensioni delle stanze. Useremo la variabile A per calcolare l'area delle stanze. Ricordate che l'area di una stanza è la sua lunghezza moltiplicata per la larghezza.

```
190 LET A=0
200 FOR I=1 TO N
210 PRINT I,
220 INPUT LU,LA
230 LET A=A+LU*LA
240 NEXT I
```

Dobbiamo ora convertire i centimetri quadri in metri quadri e poi stampare la quantità di moquette richiesta.

```
250 LET A=A/10000
255 PRINT
260 PRINT A;"METRI QUADRI RICHIESTI"
```

A questo punto possiamo mettere nel programma la tabella dei prezzi sotto forma di istruzioni DATA.

```
270 DATA 10000,8500,7250
280 DATA 13250,12000,9750
290 DATA 16000,14000,11250
300 DATA 20000,17200,15250
```

Poi possiamo far stampare l'intestazione richiesta per la stampa del prezzo.

```
310 PRINT
320 PRINT "TIPO DI MOQUETTE","    COSTO"
330 PRINT
```

Ora veniamo al punto del programma in cui sarà utile una subroutine. Dal momento che non sappiamo esattamente dove la subroutine deve cominciare, useremo semplicemente un numero di riga alto e lo correggeremo in seguito se necessario.

```
340 REM CALCOLA IL PREZZO DEL TIPO A
350 GOSUB 800
```

Ora scriviamo la subroutine. Primo, per ciascun tipo di moquette abbiamo bisogno dei tre prezzi. Possiamo fare questo leggendoli dalle istruzioni DATA.

```
800 REM SUBROUTINE DI CALCOLO DEL PREZZO
810 READ C1,C2,C3
```

Poi si controlla per vedere se l'area della moquette è meno di 15, fra 15 e 25, o più di 25 metri quadri e poi si calcola il prezzo di conseguenza.

```
820 IF A>25 THEN 860
830 IF A>15 THEN 880
840 LET P=C1*A
850 GOTO 890
860 LET P=15*C1 + 10*C2 + (A-25)*C3
870 GOTO 890
880 LET P=15*C1 + (A-15)*C2
890 RETURN
```

Studiate questo segmento di programma per convincervi che il prezzo viene calcolato esattamente. Ora possiamo tornare al programma principale e stampare il primo prezzo.

```
360 PRINT "A";TAB(25);FNR(P)
```

Una volta che questo schema è stato stabilito, il resto del programma principale segue con facilità.

```
370 REM CALCOLA IL PREZZO DEL TIPO B
380 GOSUB 800
390 PRINT "B";TAB(25);FNR(P)
400 REM CALCOLA IL PREZZO DEL TIPO C
410 GOSUB 800
420 PRINT "C";TAB(25);FNR(P)
430 REM CALCOLA IL PREZZO DEL TIPO D
440 GOSUB 800
450 PRINT "D";TAB(25);FNR(P)
460 END
```

L'istruzione END alla riga 460 è necessaria per far sì che il programma non cada nella subroutine. L'importanza della subroutine diventa evidente quando vediamo che, se non fosse stata disponibile, ciascuna delle quattro istruzioni GOSUB avrebbe dovuto essere sostituita con tante istruzioni quante ce ne sono nella subroutine.
Il programma completo è

```
100 DEF FNR(X)=INT(X/1000)*1000
110 PRINT "QUANTE STANZE";
120 INPUT N
130 PRINT "PER OGNI STANZA, IMMETTERE LUNGHEZZA"
140 PRINT "E LARGHEZZA IN CENTIMETRI"
150 PRINT "SEPARATE DA UNA VIRGOLA"
160 PRINT
170 PRINT "STANZA","DIMENSIONI"
180 PRINT
190 LET A=0
200 FOR I=1 TO N
210 PRINT I,
220 INPUT LU,LA
230 LET A=A+LU*LA
240 NEXT I
250 LET A=A/10000
255 PRINT
260 PRINT A;"METRI QUADRI RICHIESTI"
270 DATA 10000,8500,7250
280 DATA 13250,12000,9750
290 DATA 16000,14000,11250
300 DATA 20000,17200,15250
310 PRINT
320 PRINT "TIPO DI MOQUETTE","    COSTO"
330 PRINT
340 REM CALCOLA IL PREZZO DEL TIPO A
350 GOSUB 800
360 PRINT "A";TAB(25);FNR(P)
370 REM CALCOLA IL PREZZO DEL TIPO B
380 GOSUB 800
390 PRINT "B";TAB(25);FNR(P)
400 REM CALCOLA IL PREZZO DEL TIPO C
410 GOSUB 800
420 PRINT "C";TAB(25);FNR(P)
430 REM CALCOLA IL PREZZO DEL TIPO D
440 GOSUB 800
450 PRINT "D";TAB(25);FNR(P)
460 END
800 REM SUBROUTINE DI CALCOLO DEL PREZZO
810 READ C1,C2,C3
820 IF A>25 THEN 860
830 IF A>15 THEN 880
840 LET P=C1*A
850 GOTO 890
```

```
860 LET P=15*C1 + 10*C2 + (A-25)*C3
870 GOTO 890
880 LET P=15*C1 + (A-15)*C2
890 RETURN
```

9.5 Problemi

1. Studiate il programma qui sotto e scrivete quello che stamperebbe qualora fosse eseguito.

```
100 DEF FNA(X)=2+X
110 DEF FNB(Y)=10*Y
120 DEF FNC(Z)=Z↑2
130 LET R=2
140 LET S=3
150 LET T=5
160 PRINT FNC(T);FNA(S);FNB(R)
170 LET R=S+T
180 PRINT FNA(R) + FNB(S) + FNC(T)
190 END
```

2. Che cosa verrà stampato se si esegue il programma seguente?

```
100 DEF FNX(A)=6*A
110 DEF FNY(B)=B+10
120 DEF FNZ(C)=C↑3
130 READ P,Q,R
140 DATA 1,1,3
150 PRINT FNX(R);FNZ(P);FNY(Q)
160 PRINT FNY(P+Q) + FNX(R)
170 END
```

3. L'area di un cerchio è π volte il raggio (R) al quadrato, e il volume della sfera è 4/3 per π volte R al cubo. π è 3.14159 e R è il raggio del cerchio o il raggio della sfera. Definire due istruzioni DEF, una per il cerchio e una per la sfera. Costituire un'iterazione FOR NEXT su R per R che varia da 1 a 5 con incrementi di 0.5. Usare le funzioni definite per far stampare una tabella delle aree e dei volumi per ciascuno dei valori di R.

4. Che cosa si avrebbe in uscita se il programma che segue venisse eseguito?

```
100 DIM A(5)
110 FOR I=1 TO 5
120 READ A(I)
130 NEXT I
140 DATA 6,2,7,1,3
150 GOSUB 500
160 LET A(3)=10
170 GOSUB 500
180 LET A(5)=8
190 GOSUB 500
200 END
500 FOR I=1 TO 4
510 LET A(I)=A(I+1)
520 NEXT I
530 GOSUB 600
540 RETURN
600 FOR J=1 TO 5
610 PRINT A(J);
620 NEXT J
630 RETURN
```

5. Che cosa verrà stampato quando sarà eseguito il programma seguente?

```
100 LET X=10
110 GOSUB 500
120 PRINT S
130 LET X=X/2
140 GOSUB 500
150 PRINT S
160 LET X=X+3
170 GOSUB 500
180 PRINT S
190 END
500 LET S=0
510 FOR Y=1 TO X
520 LET S=S + Y
530 NEXT Y
540 RETURN
```

6. Si presuma che un array contenga dei numeri da sommare fra loro. Il primo elemento dell'array Z(1) dà il numero degli elementi che seguono nell'array e che devono essere sommati. Scrivere una subroutine che inizi dalla riga 800 per calcolare la somma degli elementi a partire da Z(1). Assegnare la somma alla variabile T. Far terminare la subroutine con un'istruzione RETURN. Presumiamo che l'array Z sia stato dimensionato in modo appropriato e che i valori dell'array siano stati caricati nel programma principale.

7. X è un array monodimensionale. Il primo elemento dell'array X(1) dà il numero degli elementi di dati che seguono nell'array. Scrivere una subroutine che inizi alla riga 500 per cercare nell'array il valore più grande. Assegnare questo valore alla variabile L. Terminare la subroutine con un'istruzione RETURN. Supponete che l'array X sia stato dimensionato correttamente e caricato con i numeri altrove.

8. Supponete che come parte dei dati in uscita si abbia bisogno di una serie di quaranta * in un'unica riga nella pagina. Per fare questo scrivere una subroutine che inizi alla riga 1000. Far terminare la subroutine con un'istruzione RETURN.

9. Supponete che un array monodimensionale Y sia stato caricato con dei numeri. Il primo elemento Y(1) dà il numero degli elementi che seguono. Vogliamo una subroutine per calcolare la media (M) e la deviazione standard (S) dei numeri dell'array che seguono il primo elemento. Iniziate la subroutine alla riga 900 e fatela terminare con un'istruzione RETURN. Le formule per i calcoli della media e della deviazione standard sono fornite qui di seguito.

   Media = (Somma dei valori)/N

$$\text{Deviazione standard} = \sqrt{\frac{N \times (\text{somma dei quadrati dei valori}) - (\text{somma dei valori})^2}{N \times (N-1)}}$$

9.6 Test di apprendimento

1. Se DEF FNA(X) = SQR(X)+3*X, Z = 2.5 e W = 10, che cos'è

   a. FNA(1)

   ....

   b. FNA(4)

   ....

   c. FNA(9)

   ....

d. FNA(Z * W)

....

2. Che cosa si avrà in uscita se viene eseguito il seguente programma?

```
100 DEF FNR(X)=X*X
110 DEF FNS(X)=3*X
120 DEF FNT(Y)=Y+1
130 LET A=1
140 PRINT FNT(A);FNR(A);FNS(A)
150 LET M=4
160 PRINT FNR(SQR(M))
170 END
```

3. Riguardo alle subroutine:

   a. Come si passa il controllo dal programma principale alla subroutine?

   ....

   b. Come si passa il controllo dalla subroutine di nuovo al programma principale?

   ....

4. Che cosa verrà stampato se eseguiamo il seguente programma?

```
100 LET A=1
110 GOSUB 200
120 LET A=A+4
130 GOSUB 200
140 LET A=A-2
150 GOSUB 200
160 END
200 REM SUBROUTINE
210 IF A<2 THEN 250
220 IF A=3 THEN 270
230 PRINT "ROSSO"
240 GOTO 280
250 PRINT "BIANCO"
260 GOTO 280
270 PRINT "BLU"
280 RETURN
```

....

Capitolo

La grafica e il colore 10

10.1 Obiettivi

CONTROLLO DEI PROGRAMMI

Impareremo ulteriori tecniche di controllo dei programmi che renderanno più facile la programmazione grafica.

POSIZIONAMENTO DEI CARATTERI

Per la programmazione grafica dovrete essere in grado di posizionare un carattere ovunque sullo schermo. Imparerete ad usare un'apposita subroutine.

LE POSSIBILITÀ GRAFICHE DEL C-64

Il C-64 ha un interessante set di caratteri grafici e di modi per visualizzarli. Imparerete l'uso del modo inverso e di alcuni caratteri grafici.

CONTROLLO DEL COLORE

L'uso del colore sul video rende le presentazioni più gradevoli e interessanti. Imparerete a scegliere il colore di ogni carattere stampato sullo schermo.

ESEMPI DI PROGRAMMI

Applicherete ciò che avete imparato per esplorare le capacità grafiche a colori del C-64.

10.2 Esercizi di scoperta

1. Accendete il computer e la TV. Inserite il seguente programma.

```
100 INPUT A
200 PRINT "A",A=1,A=2,A=3
300 GOTO 100
400 END
```

2. Eseguite il programma. Alla richiesta di input battete 2. Qualcuno dei tre numeri che appaiono è uguale a -1?

   ....

   In corrispondenza del numero 2 che inserite per A quale delle tre affermazioni A=1, A=2, A=3 è vera?

   ....

   Se all'input inserite 3, quale sequenza di numeri appare sullo schermo?

   ....

   Inserite 3 e verificate la risposta.

   ....

3. Interrompete il programma e aggiungete le seguenti righe:

```
110 PRINT TAB(10);1,2,3
120 PRINT
210 PRINT
220 PRINT "B",B=1,B=2,B=3
250 PRINT
```

4. Ora cambiate così la riga 100:

```
100 INPUT A,B
```

Listate il programma che dovrebbe apparire in questa forma:

```
100 INPUT A,B
110 PRINT TAB(10);1,2,3
120 PRINT
200 PRINT "A",A=1,A=2,A=3
210 PRINT
220 PRINT "B",B=1,B=2,B=3
250 PRINT
300 GOTO 100
400 END
```

Eseguitelo. Alla richiesta di input battete

1,1
2,3
3,3

Per quale coppia di numeri i −1 sono nella stessa colonna?

....

5. Interrompete il programma. Aggiungete le righe 230 e 240:

```
230 IF A=1 AND B=1 THEN PRINT
"CONDIZIONE VERIFICATA":PRINT:GOTO 100
240 PRINT "CONDIZIONE NON VERIFICATA"
```

Visualizzate il programma e osservate i due punti prima e dopo il secondo PRINT alla fine della riga 230. In questa istruzione vengono eseguiti due PRINT e un GOTO se la condizione è verificata. Se eseguite il programma, quale coppia di numeri deve essere inserita per far apparire CONDIZIONE VERIFICATA?

....

Eseguite il programma e verificate la vostra risposta.

6. Interrompete il programma. Cambiate la riga 230 come segue:

```
230 IF A=1 OR B=1 THEN PRINT
"CONDIZIONE VERIFICATA":PRINT:GOTO 100
```

Visualizzate il programma. Quali coppie di numeri faranno apparire CONDIZIONE VERIFICATA?

....

Provate le vostre coppie di numeri e guardate se avevate ragione. Avreste dovuto scrivere 5 coppie di numeri. Se l'input sia di A che di B è 1, la condizione è verificata?

....

7. Passiamo ad un altro argomento. Cancellate la memoria e battete il seguente programma.

```
100 GET A$
110 IF A$="" THEN 100
120 PRINT ASC(A$)
130 IF A$<>"Q" THEN 100
140 END
```

Riempite la seguente tabella premendo le combinazioni dei tasti indicate.

Combinazione dei tasti	Risultato	Codice ASCII
SHIFT/CLR/HOME	Pulisce lo schermo	______
CLR/HOME	Posiziona il cursore in HOME (nell'angolo in alto a sinistra dello schermo)	______
SHIFT/ ⇐ CRSR ⇒	Cursore verso sinistra	______
⇐ CRSR ⇒	Cursore verso destra	______
SHIFT/ ⇑ CRSR ⇓	Cursore in alto	______
⇑ CRSR ⇓	Cursore in basso	______

8. Riempite la seguente tabella premendo il tasto CTRL ed i numeri indicati.

Combinazione dei tasti	Risultato	Codice ASCII
CTRL/1	BLK (nero)	______
CTRL/2	WHT (bianco)	______
CTRL/3	RED (rosso)	______

Combinazione dei tasti	Risultato	Codice ASCII
CTRL/4	CYN (cyan)	________
CTRL/5	PUR (viola)	________
CTRL/6	GRN (verde)	________
CTRL/7	BLU (blu)	________
CTRL/8	YEL (giallo)	________
CTRL/9	Reverse ON	________
CTRL/0	Reverse OFF	________

9. Interrompete il programma e battete

```
PRINT "
```

Ora premete ciascuno dei tasti indicati ai punti 7 e 8. Notate che i caratteri vengono visualizzati. Ora premete RETURN. Cosa accade?

....

Con CHR$ e il codice ASCII potete usare, se volete, la combinazione dei tasti desiderata. Per esempio, battete

```
PRINT " SHIFT/CLEAR/HOME "
```

Cosa succede?

....

Premete RUN/STOP/RESTORE per tornare ai caratteri standard. Ora battete

```
PRINT CHR$(147)
```

Ricordate, come avete visto nella tabella al punto 7, che il valore ASCII di SHIFT/CLR/HOME è 147. Si è pulito anche questa volta lo schermo?

....

10. Ora passiamo alla programmazione grafica. Cancellate la memoria. Ricordatevi che SHIFT/ ⇑CRSR⇓ ha codice ASCII 17, ⇐CRSR⇒ ha codice

ASCII 29 e CLR/HOME 19. Battete la seguente subroutine:

```
1000 REM SUBROUTINE DI POSIZIONAMENTO DEL CURSORE
1010 IF R<1 OR R>25 OR C<1 OR C>40 THEN END
1020 IF R=25 AND C=40 THEN END
1030 PRINT CHR$(19);
1040 IF R=1 THEN 1080
1050 FOR J=1 TO R-1
1060 PRINT CHR$(17);
1070 NEXT J
1080 IF C=1 THEN 1120
1090 FOR K=1 TO C-1
1100 PRINT CHR$(29);
1110 NEXT K
1120 RETURN
```

Controllate attentamente la subroutine, in modo particolare i punti e virgola alla fine delle righe 1030, 1060, 1100. Questa subroutine usa i valori R e C per posizionare il cursore, così che il carattere successivo verrà messo sulla riga R e sulla colonna C. I codici ASCII dei tasti del cursore sono usati per spostare il cursore alla posizione desiderata. Le righe 1010 e 1020 controllano che il cursore non sia uscito dallo schermo.

11. Ora battete un programma che utilizza questa subroutine:

```
100 PRINT CHR$(147)
110 PRINT CHR$(19);TAB(10);"RIGA E COLONNA";
120 INPUT R,C
130 GOSUB 1000
140 PRINT "X";
150 GOTO 110
160 END
```

Visualizzatelo ed eseguitelo. Alle richieste di input immettete le seguenti coppie di numeri:

1,1
5,7
5,8
5,40
25,5
25,39
25,40

Quanti caratteri è largo lo schermo?

....

Quanti caratteri è alto lo schermo?

....

Con questa subroutine riuscite a mettere una X nella 25ª riga e nella 40ª colonna?

....

12. Ora cambiamo la X in un carattere grafico del C-64. Correggete la riga 140 così:

```
140 PRINT CHR$(122)
```

Eseguite il programma e alla richiesta di input immettete le seguenti coppie di numeri:

13,20
20,5

Disegnate il simbolo che è apparso sullo schermo.

....

13. Interrompete il programma. Battete la riga 140 come segue

```
140 PRINT "SHIFT/Z "
```

Notate che il rombo è sulla destra della faccia anteriore del tasto Z. Eseguite il programma e alla richiesta di input immettete 13,20. Il simbolo che appare è uguale?

....

14. Ora daremo un'occhiata ai caratteri inversi del C-64. Cambiate la riga 140 così:

```
140 PRINT CHR$(18);CHR$(122);
```

Eseguite il programma immettendo 13,20 alla richiesta di input. Il rombo è cambiato?

....

15. Infine cambiate la riga 140 così

```
140 PRINT CHR$(18);" ";
```

Comparirà uno spazio in modo inverso. Eseguite il programma battendo 13,20 alla richiesta di input. Come appare lo spazio inverso?

....

16. Se non avete una TV a colori, saltate al punto 21.

17. Cancellate dalla memoria il programma con la sua subroutine. Battete il seguente programma:

```
100 REM CODICI ASCII PER I COLORI
110 DATA NERO,144,BIANCO,5,ROSSO,28
120 DATA CYAN,159,VIOLA,156,VERDE,30
130 DATA BLU,31,GIALLO,158
140 FOR K=1 TO 8
150 READ COLR$,COLR
160 PRINT CHR$(COLR);
170 FOR I=1 TO 80
180 PRINT CHR$(18);" ";
190 NEXT I
200 NEXT K
210 END
```

Visualizzatelo ed eseguitelo. Dovreste vedere delle strisce di colore sullo schermo nell'ordine dato dalle righe 110, 120 e 130. Regolate il video in modo che i colori corrispondano ai loro nomi il più possibile. CHR$ (codice ASCII di un colore) stampa sullo schermo i caratteri di quel colore. Notate che READY si legge difficilmente se è visualizzato in giallo su fondo blu chiaro. Per rendere lo schermo più leggibile, premete RUN/STOP e RESTORE per far tornare il video ai colori di partenza.

18. Cancellate la memoria e battete il seguente programma:

```
100 REM CODICI ASCII PER I COLORI
110 DATA GIALLO,158,VERDE,30,ROSSO,28
120 FOR K=1 TO 3
130 READ C$(K),C(K)
140 NEXT K
150 PRINT "COLORE";
160 INPUT COLR$
170 FOR K=1 TO 3
```

```
180 IF ASC(COLR$)=ASC(C$(K)) THEN 200
190 NEXT K
200 PRINT "NUMERO DEI QUADRATI";
210 INPUT N
220 PRINT CHR$(C(K));
230 FOR I=1 TO N
240 PRINT CHR$(18);" ";CHR$(146);
250 NEXT I
260 PRINT CHR$(154);
270 PRINT
280 GOTO 150
290 END
```

Studiate il programma. Nella riga 260, CHR$(154) riporta al colore di partenza. Nella riga 180, quanti caratteri della parola immessa vengono considerati dalle funzioni ASC (vedi il punto 20 del Cap. 8)?

....

Eseguite il programma. Alla richiesta di input immettete:

```
GIALLO        ROSSO
5             51

VERDE
14
```

Cosa è successo?

....

19. Ora battete:

```
NERO
25
```

Di che colore è la striscia orizzontale?

....

20. Si concludono qui gli esercizi di scoperta. Spegnete il computer e la TV.

....

10.3 Analisi

CONTROLLO DEI PROGRAMMI

Espressioni logiche (vero/falso) come A=5 sono valutate dal BASIC C-64 come −1 o 0. Il valore A=5 dipende dal fatto che l'affermazione "A è uguale a 5" sia vera o falsa. Quando la variabile numerica A è 5, l'affermazione è vera e il valore di A=5 è −1. Quando A non è 5 l'affermazione è falsa e il valore di A=5 è 0. Il costrutto IF THEN contiene queste espressioni logiche.
Nel Capitolo 5 avete imparato l'uso di IF THEN con una sola condizione. In questo capitolo avete usato condizioni contenenti due semplici espressioni logiche separate da AND o OR. Per esempio la condizione in

```
100 IF A=5 OR B=6 THEN PRINT "OK"
```

è soddisfatta se il valore di A è 5 o il valore di B è 6. La condizione è soddisfatta anche se A=5 e B=6. Usando AND alla riga 100 si avrà:

```
100 IF A=5 AND B=6 THEN PRINT "OK"
```

Qui la condizione è soddisfatta e viene visualizzato OK solo quando il valore di A è 5 e, allo stesso tempo, il valore di B è 6.
Si possono creare condizioni più complicate. Per esempio la condizione

```
20 IF (R=5 OR C=5) AND D>7 THEN 50
```

è soddisfatta quando R o C è uguale a 5 e D è maggiore di 7.
Considerate un'espressione aritmetica del tipo (A+B)*C che contiene le operazioni aritmetiche usando le regole apprese nel Capitolo 3. Considerate una espressione logica che contiene le parole o operazioni logiche AND e OR che usano i significati logici standard. In una espressione aritmetica, il valore può essere un numero fra tanti, a seconda del valore delle variabili. Invece i valori di un'espressione logica possono essere solo −1 (vero) o 0 (falso).
I due punti (:) possono essere usati per separare istruzioni multiple sulla stessa riga. Avete usato questa possibilità solo dopo THEN in IF THEN. Per esempio, le righe del programma:

```
100 IF A=5 THEN PRINT "A=5 E' VERA":GOTO 300
200 PRINT "A=5 E' FALSA"
```

se la condizione è soddisfatta visualizzeranno A=5 È VERA e passeranno

alla riga 300. Se invece non è soddisfatta verrà stampato A=5 È FALSA. Un uso eccessivo di questa caratteristica però, può portare a programmi complicati e difficili da correggere.

POSIZIONAMENTO DEI CARATTERI

Avete usato la seguente subroutine che posiziona i caratteri sullo schermo.

```
1000 REM SUBROUTINE DI POSIZIONAMENTO DEL CURSORE
1010 IF R<1 OR R>25 OR C<1 OR C>40 THEN END
1020 IF R=25 AND C=40 THEN END
1030 PRINT CHR$(19);
1040 IF R=1 THEN 1080
1050 FOR J=1 TO R-1
1060 PRINT CHR$(17);
1070 NEXT J
1080 IF C=1 THEN 1120
1090 FOR K=1 TO C-1
1100 PRINT CHR$(29);
1110 NEXT K
1120 RETURN
```

Questa subroutine usa i valori di R e C per determinare la posizione di riga e colonna in cui si vuole mettere il carattere. La riga 1010 controlla che la posizione sia sullo schermo. La 1020 assicura che non sia usata l'ultima riga né l'ultima colonna. Se viene stampato un carattere nell'ultima posizione dello schermo il BASIC C-64 provoca un RETURN che sposta tutti i caratteri visualizzati in su di una riga. La riga 1030 usa il codice ASCII CHR$(19) per CLR/HOME, per muovere il cursore alla posizione HOME in alto a sinistra dello schermo. Le righe dalla 1050 alla 1070 e dalla 1090 alla 1110 spostano il cursore a destra del numero di spazi desiderato, CHR$(17), e in basso CHR$(29) per posizionare il cursore alla riga R e colonna C.
Le righe 1040 e 1080 si occupano dei casi speciali, quando R è 1 o C è 1. In questi casi non dovrebbero essere visualizzati sullo schermo i caratteri ⇐CRSR⇒ o ⇑CRSR⇓, perché il cursore inizia sempre alla posizione HOME nella 1ª riga e 1ª colonna.
Potete cambiare le istruzioni END nella 1010 e 1020 a seconda dei risultati che desiderate.

LE POSSIBILITÀ GRAFICHE DEL C-64

I caratteri possono essere stampati in azzurro su fondo blu o, nel modo inverso, blu su fondo azzurro. Il modo inverso si attiva con CTRL/9 o CHR$(18) e si disattiva con CTRL/0 o CHR$(146). Abbiamo già visto questi codici ASCII al punto 8 degli esercizi di scoperta. Avrete notato che i tasti 0 e 9 hanno sulla faccia anteriore impresso RVS/OFF e RVS/ON.
Il modo inverso può essere usato per stampare sullo schermo spazi inversi o quadrati. Per esempio, la riga in modo diretto

```
PRINT CHR$(18);" "
```

stamperà un quadrato. Il modo inverso viene disattivato alla fine di ogni riga PRINT.
Per creare un quadrato lampeggiante dovete attivare e disattivare il modo inverso. Si ottiene ciò battendo un quadrato, tornando indietro di uno spazio con il cursore a sinistra, CHR$(157), e poi stampando uno spazio normale. Per esempio, il programma:

```
10 PRINT CHR$(18);" ";
20 PRINT CHR$(157);
30 PRINT CHR$(146);" ";
40 PRINT CHR$(157);
50 GOTO 10
60 END
```

determina un quadrato lampeggiante quando viene eseguito. Le righe 20 e 40 muovono indietro il cursore di uno spazio. La riga 30 disattiva il modo inverso e stampa uno spazio. Notate che i punti e virgola alla fine delle righe da 10 a 40 impediscono al cursore di scendere alla riga sotto.
Il Commodore 64 ha una serie di caratteri grafici che possono essere usati per visualizzare diagrammi e figure. I caratteri grafici sono indicati a coppie sul davanti di alcuni tasti. Nel modo maiuscolo/grafico si accede ai caratteri di sinistra usando il tasto C=, a quelli di destra usando SHIFT. Ogni tasto grafico è collegato a un codice ASCII. L'elenco dei codici ASCII si trova nell'appendice C del manuale del C-64.

CONTROLLO DEL COLORE

Per cambiare il colore di un carattere sullo sfondo standard blu, si usano i tasti da CTRL/1 a CTRL/8. Sulla faccia anteriore dei tasti numerici è indicato il colore associato a ciascun tasto. La seguente tabella riporta ciascun colore, il suo tasto e il codice ASCII.

Colore	Tasto CTRL	Codice ASCII
Nero	1	144
Bianco	2	5
Rosso	3	28
Cyan	4	159
Viola	5	156
Verde	6	30
Blu	7	31
Giallo	8	158

La combinazione di inizio si ottiene premendo RUN/STOP/RESTORE.
La seguente istruzione in modo diretto visualizza un quadrato verde:

```
PRINT CHR$(18);CHR$(30);" "
```

Una volta immessa, tutti i caratteri battuti in seguito appariranno verdi su fondo blu.
Ulteriori informazioni su come cambiare il colore dello sfondo e della cornice e sugli altri otto colori disponibili sul C-64 sono contenute nel Capitolo 3 della *Guida di riferimento per il programmatore*.

10.4 Esempi di programmi

ESEMPIO 1 — MOVIMENTO DEL CURSORE

Questo programma vi permette di muovere un quadrato sullo schermo con i tasti U, D, R e L. Useremo la stessa subroutine già usata negli esercizi di scoperta.
Ricordate che CHR$(147) pulisce lo schermo, CHR$(18) attiva il modo inverso, CHR$(146) lo disattiva e CHR$(157) muove il cursore indietro di uno spazio. Segue il programma completo:

```
100 PRINT CHR$(147)
110 LET R=13
120 LET C=20
130 GOSUB 1000
140 PRINT CHR$(18);" ";
150 GET A$
160 IF A$="" THEN 150
170 IF A$="U" THEN R=R-1
180 IF A$="D" THEN R=R+1
190 IF A$="R" THEN C=C+1
```

```
200 IF A$="L" THEN C=C-1
210 IF R<1 OR R>25 THEN 100
220 IF C<1 OR C>40 THEN 100
230 PRINT CHR$(157);CHR$(146);" ";
240 GOSUB 1000
250 PRINT CHR$(18);" ";
260 GOTO 150
270 END
1000 REM SUBROUTINE DI POSIZIONAMENTO DEL CURSORE
1010 IF R<1 OR R>25 OR C<1 OR C>40 THEN END
1020 IF R=25 AND C=40 THEN END
1030 PRINT CHR$(19);
1040 IF R=1 THEN 1080
1050 FOR J=1 TO R-1
1060 PRINT CHR$(17);
1070 NEXT J
1080 IF C=1 THEN 1120
1090 FOR K=1 TO C-1
1100 PRINT CHR$(29);
1110 NEXT K
1120 RETURN
```

Le righe 100-140 cancellano lo schermo e mettono un quadrato al centro dello schermo in riga 13 e colonna 20. Le righe 150 e 160 controllano se è stato premuto un tasto. Le righe 170-200 stabiliscono la posizione in cui sarà stampato il quadrato cambiando i valori di R e C. La riga 230 cancella il quadrato e le righe 240 e 250 producono un nuovo quadrato nella nuova posizione.
Eliminando la riga 130 potete usare questo programma per tracciare linee e figure.

ESEMPIO 2 — GRAFICI A BARRE

Il seguente programma legge i nomi e le votazioni di ciascun studente di una classe. I nomi sono visualizzati insieme a un grafico a barre sulla destra, a partire dalla posizione 10. Il numero di quadrati nella barra corrisponde alla votazione. Segue il programma completo:

```
100 PRINT CHR$(147)
110 READ N
120 FOR J=1 TO N
130 READ NOME$,RISUL
140 PRINT NOME$;TAB(10);
150 REM DISEGNA UN GRAFICO A BARRE
160 FOR K=1 TO RISUL
170 PRINT CHR$(18);" ";
180 NEXT K
```

```
190 PRINT
200 PRINT
210 NEXT J
220 DATA 3
230 DATA ALDO,4
240 DATA ADA,7
250 DATA MARA,8
260 END
```

ESEMPIO 3 — GRAFICI A BARRE COLORATE

Il programma precedente può essere modificato per disegnare barre colorate. Per far questo si aggiungono le righe da 30 a 90, la 165, 215 e si cambiano la 130, 170 e 230-250. Il programma completo è:

```
30 REM CARICA I CODICI DEI COLORI NELL'ARRAY C
40 FOR I=1 TO 3
50 READ C(I)
60 NEXT I
70 REM CODICI DEI COLORI VERDE,ROSSO,GIALLO
80 DATA 30,28,158
90 REM PULISCE LO SCHERMO
100 PRINT CHR$(147)
110 READ N
120 FOR J=1 TO N
130 READ NOME$,RISUL,COL
140 PRINT NOME$;TAB(10);
150 REM DISEGNA UN GRAFICO A BARRE
160 FOR K=1 TO RISUL
165 REM C(COL) E' 28,30, O 158
170 PRINT CHR$(C(COL));CHR$(18);" ";CHR$(154);
180 NEXT K
190 PRINT
200 PRINT
210 NEXT J
215 REM NOME,RISUL E COL(1,2,3)
220 DATA 3
230 DATA ALDO,4,1
240 DATA ADA,7,2
250 DATA MARA,8,3
260 END
```

Le righe 30-70 caricano nell'array numerico C i codici dei colori, dati nella riga 80. C(1) è il codice colore per il verde; C(2) per il rosso, C(3) per il giallo. Nella riga 170, il codice del blu chiaro, colore di partenza, CHR$(154) viene stampato alla fine di ogni blocco.
I nomi sono così stampati nel colore standard. L'informazione sul colore

(1,2,3) è aggiunta alle righe 230-250. Cambiando le istruzioni DATA dalla linea 220 si può modificare il colore delle barre.

ESEMPIO 4 — ANIMAZIONE

Questo programma fa muovere un quadrato sullo schermo come se fosse animato. Troverete i caratteri grafici per gli angoli sui tasti A, S, Z e X e il trattino sul tasto C. Ai caratteri grafici per gli angoli si accede mediante il tasto C=. Il trattino è invece ottenibile mediante SHIFT nel modo standard (maiuscolo/grafico). Il trattino verticale si ottiene battendo SHIFT/B. La parte della riga 60 tra virgolette si ottiene battendo C=/**A**, SHIFT/C tre volte e C=/S. Segue il programma completo. Sullo schermo i trattini saranno uniti e allineati, a differenza di ciò che vedete qui.

```
100 PRINT CHR$(147);
110 LET J=1
120 PRINT CHR$(19)
130 FOR I=1 TO 5
140 PRINT
150 NEXT I
160 PRINT TAB(J);"┌───┐"
170 PRINT TAB(J);"│   │"
180 PRINT TAB(J);"│   │"
190 PRINT TAB(J);"└───┘"
200 LET J=J+1
210 IF J>33 THEN END
220 PRINT CHR$(147);
230 GOTO 120
240 END
```

La riga 100 cancella lo schermo. Le righe 120-150 muovono il cursore in giù di cinque righe. Le righe 160-190 stampano il quadrato a partire dalla colonna J che è inizialmente posto a 1 nella riga 110. La riga 200 incrementa il valore di J di 1 e la 220 controlla che J non sia troppo grande per stampare il quadrato correttamente. Le righe 220 e 230 cancellano lo schermo e il quadrato e fanno reiniziare il processo. Usando i caratteri grafici nelle righe 160-190 si possono creare altre figure.

10.5 Problemi

1. Cosa visualizzerà il seguente programma?

```
100 LET T=2
110 LET S=7
```

```
120 IF T<8 AND T>4 THEN PRINT "T OK":GOTO 140
130 IF S<1 OR S>5 THEN PRINT "S OK"
140 END
```

2. Cosa accade quando si esegue questo programma?

```
10 LET A=5
20 PRINT "A=5 E' ";A=5
30 PRINT "A=6 E' ";A=6
40 END
```

3. In che modo dovete modificare la subroutine che posiziona il cursore nel punto 10 degli esercizi di scoperta in modo che stampi un messaggio d'errore "FUORI DALLO SCHERMO" quando i valori di R e C sono troppo grandi o troppo piccoli?

4. Scrivete un programma che faccia muovere un quadrato verde su e giù sullo schermo in continuazione.

5. Scrivete un programma che faccia muovere un quadrato rosso diagonalmente sullo schermo. Il quadrato dovrebbe fermarsi prima di raggiungere il fondo dello schermo.

6. Modificate le istruzioni DATA nel programma dell'esempio 3 in modo che la prima barra sia gialla, la seconda verde e la terza rossa.

10.6 Test di apprendimento

1. Che valore di B bisogna immettere perché il programma visualizzi OK?

```
10 INPUT B
20 IF B<7 AND B>5 THEN PRINT "OK"
30 END
```

....

2. Cosa accadrà quando verrà eseguita la seguente istruzione in modo diretto ?

```
PRINT CHR$(147);
```

....

3. Cosa verrà visualizzato dalla seguente istruzione?

```
PRINT CHR$(18);"J"
```

....

4. Di che colore sarà il quadrato visualizzato dalla seguente istruzione in modo diretto?

```
PRINT CHR$(28);CHR$(18);" "
```

....

5. Cosa occorre per visualizzare il carattere grafico ├ sullo schermo? (Suggerimento: guardate sul lato anteriore dei tasti)

....

Capitolo

Numeri casuali e simulazioni 11

11.1 Obiettivi

Una delle più interessanti applicazioni dei computer riguarda la simulazione di eventi o di processi che implicano un elemento di casualità. Esempi ne possono essere l'uso del computer per simulare giochi di azzardo o per calcolare quanti dovranno essere gli impiegati di banca addetti agli sportelli per far sì che i clienti che arrivano non debbano aspettare più di pochi minuti per essere serviti. In questo capitolo vedremo come il computer può essere usato per trattare problemi di questo tipo. I nostri obiettivi sono i seguenti.

CARATTERISTICHE DEL GENERATORE DI NUMERI CASUALI

Il Commodore 64 ha una funzione generatrice di numeri casuali che è il cuore di tutti i programmi che implicano un elemento aleatorio o casuale. Impareremo come impiegarla nei programmi BASIC.

SELEZIONARE I NUMERI CASUALI

Generalmente il generatore di numeri casuali è usato per produrre gruppi di numeri casuali con caratteristiche specificate dal programmatore. Vedremo come si può ottenere questo e come possa essere generato qualunque gruppo desiderato di numeri.

ESEMPI DI PROGRAMMI

Gli esercizi di programmazione e i problemi di questo capitolo useranno simulazioni e applicazioni che comportano l'elemento casuale.

11.2 Esercizi di scoperta

IL GENERATORE DI NUMERI CASUALI

Prima di iniziare il lavoro sul computer, dobbiamo conoscere alcune importanti caratteristiche del generatore di numeri casuali. Per propria natura questi generatori producono sequenze di numeri che sembrano non avere alcuno schema o alcuna relazione fra loro. Perché un generatore di numeri casuali sia utile, è necessario che, tutte le volte che un programma lo utilizza, si ottenga una diversa sequenza di numeri. Questo però fa sorgere un problema interessante. Supponiamo che un programma che usa i numeri casuali non funzioni come dovrebbe. Se il problema ha a che fare con i numeri casuali, può essere estremamente difficile correggerlo dal momento che, ogni volta che viene eseguito il programma, vengono generati a caso dei numeri diversi. Per questo motivo, il BASIC C-64 può far ripetere una data sequenza di numeri casuali tutte le volte che il programma viene avviato. Per far uso di questa possibilità, potete dare un numero di partenza o "seme" come generatore di numeri casuali. Ricordate però che questa caratteristica del BASIC dev'essere usata solo quando state cercando degli errori in un programma.

1. Accendete il computer e la TV. A meno che sia diversamente specificato, presumete che vengano generate delle sequenze di numeri diversi tutte le volte che un programma viene eseguito.

2. Immettete il seguente programma:

```
100 FOR I=1 TO 8
110 PRINT RND(1)
120 NEXT I
130 END
```

Eseguite il programma. Quante cifre decimali ci sono nel numero più lungo?

....

Eseguite di nuovo il programma. Gli otto numeri generati, sono gli stessi di prima?

....

3. Aggiungete la riga 90:

```
90 PRINT RND(-1)
```

Eseguite il programma due volte. Le serie di numeri sono le stesse?

....

PRINT RND(−1) fa partire il generatore di numeri casuali al numero 2.99196472E-08 e lo stampa. Gli otto numeri successivi generati sono basati su questo numero. In questo modo le due serie di numeri sono uguali.

4. Cancellate il programma in memoria. Immettete il seguente programma:

```
100 LET L=.5
110 LET S=.5
120 FOR I=1 TO 100
130 LET X=RND(1)
140 IF X>L THEN 170
150 IF X<S THEN 190
160 GOTO 200
170 LET L=X
180 GOTO 200
190 LET S=X
200 NEXT I
210 PRINT "MASSIMO = ";L
220 PRINT "MINIMO = ";S
230 END
```

Questo programma esamina tutti i numeri generati dalla funzione RND e tiene conto del numero più grande e di quello più piccolo che vengono generati. Così com'è, il programma genererà 100 numeri casuali. Eseguite il programma e registrate quello che viene scritto in uscita.

....

5. Battete la riga 120 nel modo seguente:

```
120 FOR I=1 TO 1000
```

Ora il programma, genererà 1000 numeri casuali. Il programma impiegherà circa 15 secondi. Eseguite il programma e annotate quello che è stato scritto in uscita.

....

Basandovi su quello che avete visto finora, quale pensate che sia il numero più grande che possa venir generato dalla funzione RND?

....

E che ne dite del più piccolo?

....

6. Ora proseguiamo con qualche altra idea associata ai numeri casuali. Cancellate il programma in memoria e immettete il programma seguente.

```
100 FOR I=1 TO 8
110 PRINT INT(2*RND(1));
120 NEXT I
130 END
```

Eseguite il programma e registratene l'uscita.

....

Quali sono stati gli unici due numeri stampati dal calcolatore?

....

7. Cambiate la riga 110 come segue:

```
110 PRINT INT(3*RND(1));
```

Visualizzate il programma. Se il programma viene eseguito, quali numeri pensate che saranno visualizzati?

....

Eseguite il programma diverse volte.

Potete prevedere qualcosa circa la sequenza o lo schema in cui i numeri verranno presentati?

....

8. Cambiate la riga 110 così:

```
110 PRINT INT(2*RND(1)+1);
```

Che cosa pensate che farà ora il programma?

....

Eseguite il programma e registratene qui sotto l'output.

....

9. Cambiate ancora la riga 110 nel modo seguente:

```
110 PRINT INT(4*RND(1)+5);
```

Se il programma viene eseguito, che cosa pensate che si otterrà?

....

Eseguite il programma diverse volte e descrivetene l'output.

....

C'è qualche schema nell'output?

....

10. Cambiate la riga 110 nel modo seguente. Notate la mancanza del ;.

```
110 PRINT INT(30*RND(1))/10
```

Visualizzate il programma e studiatelo attentamente. Che cosa pensate che si presenterà ora in uscita?

....

Eseguite il programma e descrivetene l'uscita.

....

11. Cambiate nuovamente la riga 110 come segue:

```
110 PRINT INT(200*RND(1))/100
```

Visualizzate il programma in memoria. Che cosa pensate che accadrà se il programma venisse eseguito?

....

Controllate se avevate ragione. Eseguite il programma e registratene qui sotto il risultato.

....

12. Aggiuntete la riga 90:

```
90 PRINT RND(-1)
```

Eseguite il programma numerose volte. Le serie di numeri sono le stesse?

....

13. Con ciò è finito per questa volta il lavoro sul computer. Spegnete il computer e il televisore.

11.3 Analisi

LA FUNZIONE RND

Non entreremo nei dettagli di come i numeri casuali vengono generati. È sufficiente dire che ci sono vari metodi matematici per produrre questi numeri. Il generatore di numeri casuali è richiamato con la funzione RND. Questa funzione si usa come le altre funzioni incorporate nel BASIC studiate in precedenza, ma ne differisce per un aspetto importante: con la funzione RND sembra che non ci sia alcuno schema, né alcuna regola nella generazione dei numeri. Naturalmente, questo è proprio il punto importante della funzione. RND sta per "casuale" (random). La funzio-

ne genera numeri fra 0 e 1 a caso. Tutti i numeri in quell'intervallo hanno la stessa possibilità di essere presentati. In realtà l'intervallo dei numeri generati è da 0.0000000 a 0.9999999. Lo zero talvolta si può presentare, ma l'uno non capita mai.
Altri argomenti positivi per RND generano gli stessi numeri casuali dell'argomento 1. La funzione RND accetta anche argomenti negativi. Un argomento negativo produce lo stesso numero (tra 0 e 1) ogni volta che è usato. Un argomento negativo di RND usato in una istruzione PRINT ripone lo stesso seme nel generatore di numeri casuali determinando il numero di partenza per il calcolo dei numeri casuali.

RND genera numeri che vanno da 0 a 0.9999999

SELEZIONARE I NUMERI CASUALI

La maggior parte delle volte non vogliamo dei numeri casuali nell'intervallo prodotto dalla funzione RND. Potremmo volere dei numeri interi a caso in un certo intervallo o un gruppo di numeri casuali con un particolare gruppo di caratteristiche. Dobbiamo perciò riflettere un momento su come generare gruppi di numeri casuali con delle caratteristiche che possiamo specificare.
Cominciamo con le caratteristiche della funzione RND. RND (1) genera numeri che stanno nell'intervallo da 0 a 1, o meglio da 0 a leggermente meno di 1. Se moltiplichiamo RND (1) per N, moltiplichiamo l'intervallo della funzione per N. Così, N*RND (1) produrrà numeri casuali nell'intervallo da 0 a N. Se lo desideriamo, possiamo spostare l'intervallo (mantenendone sempre le stesse dimensioni) aggiungendo un numero. N*RND (1) +A produrrebbe dei numeri casuali nell'intervallo da A ad (A+N), o da A a leggermente meno di (A+N). Infine, se lo desideriamo, possiamo prendere la parte intera di un'espressione usando la funzione INT, per produrre degli interi casuali. Gli esempi presentati qui sotto mostrano come può essere usata la funzione RND (1).

Espressione BASIC	Risultato
5*RND (1)+10	Numeri casuali nell'intervallo da 10 a 15
INT(5*RND (1)+10)	Numeri interi casuali 10, 11, 12, 13, 14
INT(2*RND (1)+1)	Numeri interi casuali 1 e 2
100*RND (1)	Numeri casuali nell'intervallo da 0 a 100

Nei vostri studi potete esservi imbattuti nella nozione di media e di deviazione standard (vedere il problema 9 nel Capitolo 9). Possiamo usare

la funzione RND per generare numeri che sembrano essere presi da una raccolta di numeri aventi una data media e una data deviazione standard. La regola per generare questi numeri è

X = M+S((somma di 12 numeri dalla funzione RND (1))−6)

dove M e S sono rispettivamente la media e la deviazione standard desiderate. Questa è un'applicazione in cui una subroutine sarebbe molto utile. Come si è definito sopra, i valori di X appariranno come derivanti da una raccolta di numeri con media M e deviazione standard S. I valori di X possono essere usati per simulare un processo che segue una curva a campana (distribuzione di Gauss).
Una nota sulla ricerca degli errori. Nel Commodore 64 c'è un modo per eseguire un programma varie volte e ripetere la sequenza di numeri casuali che sono generati dalla funzione RND (v. il punto 12 degli esercizi di scoperta). Normalmente è consigliabile scrivere inizialmente programmi che generano la stessa sequenza di numeri casuali ogni volta che vengono eseguiti. Una volta che siete sicuri che il programma funziona bene, potete modificarlo in modo da produrre effettivi numeri casuali.

11.4 Esempi di programmi

ESEMPIO 1 — LANCIO DI MONETE

Una delle applicazioni più facili dei numeri casuali è la simulazione del lancio di una moneta. Vogliamo scrivere un programma che, quando sia eseguito, produca in uscita:

```
LANCIO    RISULTATO

1         TESTA
2         CROCE
3         CROCE
4         TESTA
```

(ecc.)

L'uscita dev'essere determinata in modo casuale per ciascun lancio della moneta, con testa e croce aventi uguale probabilità di uscire. Il programma deve stampare i risultati di dieci lanci.
La prima parte del programma genera l'intestazione e la riga vuota sottostante.

```
100 PRINT "LANCIO","RISULTATO"
110 PRINT
```

Ora dobbiamo aprire un'iterazione per generare dieci lanci della moneta.

```
120 FOR I=1 TO 10
```

Il passo successivo è di generare degli zero e degli uno a caso. Presumete che zero significhi "testa" e uno significhi "croce". Dovreste essere in grado di capire che la seguente istruzione produrrà degli zero e degli uno a caso.

```
130 LET X=INT(2*RND(1))
```

Ora analizziamo X per vedere se è capitata testa (0) o croce (1).

```
140 IF X=0 THEN 170
150 PRINT I,"CROCE"
160 GOTO 180
170 PRINT I,"TESTA"
180 NEXT I
```

Tutto quello che resta da fare ora è di mettere l'istruzione END.

```
190 END
```

Il programma completo è elencato qui sotto.

```
100 PRINT "LANCIO","RISULTATO"
110 PRINT
120 FOR I=1 TO 10
130 LET X=INT(2*RND(1))
140 IF X=0 THEN 170
150 PRINT I,"CROCE"
160 GOTO 180
170 PRINT I,"TESTA"
180 NEXT I
190 END
```

Questo è un programma utile per dimostrare come il computer possa essere istruito a produrre sequenze diverse di numeri casuali o sequenze identiche ogni volta che il programma viene eseguito. Effettuate i necessari cambiamenti nel programma aggiungendo una riga 90 per fargli ripetere la stessa sequenza continuamente (vedi il punto 12 degli esercizi di scoperta).

ESEMPIO 2 — NUMERI INTERI CASUALI

Vediamo ora come scrivere un programma BASIC per generare e stampare cinquanta numeri interi compresi nell'intervallo da 10 a 15. L'unica parte del programma che richiederà una certa riflessione è l'istruzione per generare gli interi casuali. Ci concentreremo così su quest'unica istruzione.
Ricordate che RND (1) genera numeri compresi nell'intervallo da 0 ad 1. Così, 6∗RND (1) genererà numeri nell'intervallo da 0 a 6. In realtà il limite superiore sarà 5.999999 e non 6. Usando la funzione INT possiamo convertire i numeri casuali in numeri interi. INT(6∗RND(1)) produrrà gli interi 0, 1, 2, 3, 4, 5 a caso. Ora è chiaro che, per ottenere i numeri desiderati, dobbiamo aggiungere 10. Così, l'espressione INT(6∗RND(1))+10 produrrà i numeri che vogliamo.
Una volta che siamo arrivati a creare questa riga, il programma segue facilmente.

```
100 FOR I=1 TO 50
110 LET Y=INT(6*RND(1)) + 10
120 PRINT Y,
130 NEXT I
140 END
```

ESEMPIO 3 — DISTRIBUZIONE DI NUMERI CASUALI

Supponiamo di generare una gran quantità di numeri interi a caso nell'intervallo compreso fra 1 e 10. Se il generatore di numeri casuali del computer funziona bene, dovremmo aspettarci di ottenere lo stesso numero di ciascuno degli interi. Se generassimo 1000 interi, ci aspetteremmo di avere cento 1, cento 2, e così via. Il nostro problema sarà quello di scrivere un programma BASIC per effettuare un conteggio degli interi casuali man mano che vengono generati e poi stamparne i totali. Un controllo di questi totali ci dirà se il generatore di numeri casuali del computer è più o meno buono.
Per prima cosa, pensiamo a come faremo il conteggio. Un buon modo per farlo è di usare un array monodimensionale. X(1) conterrà il numero di 1 generati, X(2) il numero di 2, e avanti di questo passo fino a X(10). Il primo compito è così quello di dimensionare l'array e di fissare tutti i valori dell'array come uguali a zero.

```
100 DIM X(10)
110 FOR I=1 TO 10
120 LET X(I)=0
130 NEXT I
```

Poi apriamo un'iterazione per generare mille numeri, generare gli interi casuali, quindi usare gli interi come indici per incrementare gli appropriati contatori nell'array.

```
140 FOR I=1 TO 1000
150 LET Y=INT(10*RND(1)) + 1
160 LET X(Y)=X(Y) + 1
170 NEXT I
```

Ora tutto quello che resta da fare è di presentare in uscita il contenuto dell'array X.

```
180 FOR J=1 TO 10
190 PRINT J,X(J)
200 NEXT J
210 END
```

Il programma completo è:

```
100 DIM X(10)
110 FOR I=1 TO 10
120 LET X(I)=0
130 NEXT I
140 FOR I=1 TO 1000
150 LET Y=INT(10*RND(1)) + 1
160 LET X(Y)=X(Y) + 1
170 NEXT I
180 FOR J=1 TO 10
190 PRINT J,X(J)
200 NEXT J
210 END
```

Può essere interessante eseguire questo programma e constatare di persona come funziona il generatore di numeri casuali. Se diminuite il numero di interi generati, l'accordo fra quanto vi aspettate e quello che in realtà ha luogo dovrebbe peggiorare. D'altra parte, se generate un numero più grande di numeri casuali, l'accordo dovrebbe migliorare. Il computer impiegherà circa 20 secondi per visualizzare i risultati.

ESEMPIO 4 — UNA PASSEGGIATA CASUALE

Questo programma simula degli spostamenti casuali su un pavimento a piastrelle. Voi camminate a casaccio, da piastrella a piastrella e potete tornare su qualche piastrella sopra la quale siete già passati.

La riga 150 assegna ad A un valore intero scelto a caso tra 1, 2, 3 e 4. Le righe dalla 160 alla 190 usano il valore di A per decidere se la piastrella sulla quale vi sposterete è sopra, sotto, a destra o a sinistra.
La riga 230 traccia sulla piastrella un quadrato.
Le righe 200 e 210 fanno ricominciare il processo se camminate fuori del pavimento piastrellato. Di nuovo la subroutine di posizionamento del cursore che inizia alla riga 1000 è usata per posizionare il cursore prima di stampare il quadrato.

```
100 PRINT CHR$(147)
110 LET R=13
120 LET C=20
130 GOSUB 1000
140 PRINT CHR$(18);" ";
150 LET A=INT(4*RND(1))+1
160 IF A=1 THEN R=R-1
170 IF A=2 THEN R=R+1
180 IF A=3 THEN C=C-1
190 IF A=4 THEN C=C+1
200 IF R<1 OR R>25 THEN 100
210 IF C<1 OR C>40 THEN 100
220 GOSUB 1000
230 PRINT CHR$(18);" ";
240 GOTO 150
250 END
1000 REM SUBROUTINE DI POSIZIONAMENTO DEL CURSORE
1010 IF R<1 OR R>25 OR C<1 OR C>40 THEN END
1020 IF R=25 AND C=40 THEN END
1030 PRINT CHR$(19);
1040 IF R=1 THEN 1080
1050 FOR J=1 TO R-1
1060 PRINT CHR$(17);
1070 NEXT J
1080 IF C=1 THEN 1120
1090 FOR K=1 TO C-1
1100 PRINT CHR$(29);
1110 NEXT K
1120 RETURN
```

ESEMPIO 5 — COLORI CASUALI

Questo programma colora un rettangolo al centro dello schermo con quadrati di posizione e colore casuale. Il cuore del programma è nelle righe 220, 230 e 240, in cui le funzioni INT definite alle righe 180, 190 e 200 sono usate per determinare il colore e la posizione di riga e colonna del quadrato. Le funzioni sono FNC, FNRIG e FNCOL. FNC dà come risultato un intero casuale fra 1 e 8, FNRIG tra 7 e 16 e FNCOL un intero ca-

suale fra 15 e 24. La riga 270 stampa il quadrato colorato sullo schermo. Il programma qui listato manca della necessaria subroutine di posizionamento del cursore che inizia alla riga 1000.

```
100 REM CARICA I CODICI DEI COLORI NELL'ARRAY C
110 FOR I=1 TO 8
120 READ C(I)
130 NEXT I
140 DATA 144,5,28,159,156,30,31,158
150 REM PULISCE LO SCHERMO
160 PRINT CHR$(147);
170 REM DEFINISCE COLORE,RIGA E COLONNA CON LA FUNZIONE
175 REM DEI NUMERI CASUALI
180 DEF FNC(I)=INT(8*RND(1))+1
190 DEF FNRIG(I)=INT(10*RND(1))+15
200 DEF FNCOL(I)=INT(10*RND(1))+7
210 REM SELEZIONA COLORE,RIGA E COLONNA CASUALI
220 LET COLR=C(FNC(1))
230 LET R=FNRIG(1)
240 LET C=FNCOL(1)
250 GOSUB 1000
260 REM STAMPA IL QUADRATO
270 PRINT CHR$(COLR);CHR$(18);" ";
280 GOTO 210
290 END
```

ESEMPIO 6 — COMPLEANNI COINCIDENTI

Supponiamo che cinquanta estranei si trovino riuniti in una stanza. Qual è la probabilità che due persone abbiano lo stesso compleanno? (Consideriamo solo il giorno dell'anno, non l'anno di nascita). Questo è un problema famoso nella teoria delle probabilità e dà risultati sorprendenti. Possiamo attaccare il problema con la seguente strategia. Generando degli interi casuali nell'intervallo da 1 a 365 possiamo simulare un compleanno per ciascuno degli estranei. Se usiamo un array monodimensionale per i compleanni via via che vengono generati, è facile controllare se vi sono compleanni identici. Cominciando con il primo compleanno B(1), controlliamo per vedere se coincide con uno dei rimanenti. Poi facciamo la stessa cosa per B(2), e via di seguito.
Per questo esempio daremo per prima cosa uno sguardo al programma completo, poi torneremo indietro e spiegheremo quello che avviene in ciascuna riga.

```
100 DIM B(50)
110 FOR I=1 TO 50
120 LET B(I)=INT(365*RND(1)) + 1
```

```
130 NEXT I
140 LET F=0
150 FOR I=1 TO 49
160 FOR J=I+1 TO 50
170 IF B(I) <> B(J) THEN 190
180 LET F=F+1
190 NEXT J
200 NEXT I
210 PRINT "LE COPPIE CON LO STESSO COMPLEANNO SONO";F
220 END
```

La riga 100 dimensiona semplicemente un array per 50 elementi. Le righe da 110 a 130 caricano nell'array dei numeri interi scelti nell'intervallo da 1 a 365, estremi compresi. Nella riga 140 si rende la variabile F uguale a 0. Useremo questa variabile per seguire il numero dei compleanni che verranno confrontati con il resto dei compleanni della lista. Siccome ci deve essere almeno un compleanno della lista con cui effettuare i confronti, il valore di I si ferma a 49. Nella riga 160 viene inserita la seconda parte del confronto.
J inizia al valore successivo dopo il valore attuale di I e va avanti per il resto della lista. Il controllo per un paio di compleanni viene effettuato nella riga 170. Se non si trova alcuna coincidenza, si salta al successivo valore di J. Se viene trovata la coincidenza, il contatore di coppie viene incrementato di 1 nella riga 180. I risultati vengono stampati nella riga 210. Un problema che sorge con questo programma è che registrerebbe tre persone con lo stesso compleanno come due coppie con lo stesso compleanno. Riuscite a immaginare un modo per rimediare a questo?
Con questo programma potete fare interessanti esperimenti. Il numero delle persone può essere cambiato con semplici modifiche del programma. Il programma può essere eseguito molte volte per vedere quante coppie di compleanni in media si trovano in una folla di una certa dimensione.

11.5 Problemi

1. Scrivere un programma per generare e presentare in uscita venticinque numeri casuali come 5.3 nella forma X.Y dove X e Y sono cifre scelte a caso dal gruppo 0, 1, 2, 3 ... 9.

2. Scrivere un programma per generare e stampare cinquanta interi scelti a caso dall'intervallo 13 ~ 25.

3. Che cosa verrà presentato in uscita se sarà eseguito il seguente programma?

```
100 FOR N=1 TO 20
110 PRINT INT(20*RND(1)+1)/100
120 NEXT N
130 END
```

4. Se il programma seguente venisse eseguito, che cosa verrebbe stampato?

```
100 FOR I=1 TO 10
110 PRINT INT(100*RND(1))/10
120 NEXT I
130 END
```

5. Scrivere un programma che simuli il lancio di una moneta 10, 50, 100, 500 e 1000 volte. In ciascun caso, stampare il numero totale di risultati (testa o croce).

6. Costruire una simulazione del lancio dei dadi in BASIC. I dadi devono essere lanciati venti volte. Per ciascun lancio, presentare le facce dei dadi che sono rivolte in alto.

7. Scrivere un programma per generare e stampare la media di 100 numeri casuali scelti nell'intervallo da 0 a 1.

8. Modificare il programma dell'esempio 5 ed eseguirlo tante volte quante sono necessarie per trovare le dimensioni di una folla in cui ci sia almeno il 50 per cento di probabilità che due persone abbiano lo stesso compleanno.

9. Giovanni e Mario vogliono incontrarsi in biblioteca. Si mettono d'accordo di arrivare in biblioteca fra l'una e le due del pomeriggio. Si accordano poi che aspetteranno dieci minuti dopo essere arrivati (ma non dopo le due), e se l'altro non è arrivato se ne andranno. Scrivere un programma BASIC per calcolare la probabilità che Giovanni e Mario hanno di incontrarsi. Fate una simulazione del problema usando il generatore di numeri casuali.

10. Supponiamo che una scatola contenga delle palline colorate. Ci sono dieci palline rosse, cinque blu, due verdi e undici gialle. Scrivere un programma BASIC che simuli l'estrazione di cinque palline a caso dalla scatola; quelle estratte in sequenza non sono più rimesse nella scatola.

11. Usare la regola data nella sezione Analisi di questo capitolo per gene-

rare e stampare venticinque numeri scelti a caso da una distribuzione gaussiana di numeri con media 10 e deviazione standard. Arrotondare poi i numeri a due decimali.

12. Supponiamo che un fabbricante di sapone decida di scegliere un nome di cinque lettere per il suo prodotto. Il primo, il terzo e il quinto carattere sono scelti a caso dalle lettere BCDFGHJKLMNOPQRSTV WXYZ. La seconda e la quarta lettera sono scelte a caso fra le vocali AEIOU. Scrivere un programma per generare e presentare in uscita cento prove per il nome del sapone usando le regole suddette.

13. Modificare il programma del problema 6 per simulare il lancio di un paio di dadi per 1000 volte. Visualizzare il numero di volte che ciascuna delle undici possibili combinazioni capita nella simulazione.

11.6 Test di apprendimento

1. Scrivere un programma per generare e stampare 100 numeri interi casuali dal gruppo 1, 2, 3 e 4.

2. Scrivere un programma per generare e presentare in uscita 100 numeri casuali che stiano nell'intervallo da 25 a 50.

3. Che cosa verrà stampato se si esegue il seguente programma?

```
100 FOR I=1 TO 10
110 LET N=INT(2*RND(1)+1)
120 IF N=1 THEN 150
130 PRINT "BIANCO"
140 GOTO 160
150 PRINT "ROSSO"
160 NEXT I
170 END
```

....

4. Che cosa verrà stampato se si esegue il seguente programma?

```
100 FOR J=1 TO 5
110 PRINT INT(1000*RND(1))/100
120 NEXT J
130 END
```

....

Capitolo

I file 12

12.1 Obiettivi

Potete usare file impostati su dischetto per immagazzinare e richiamare serie di informazioni. In questo capitolo imparerete a usare i file per manipolare le informazioni.

APRIRE E CHIUDERE I CANALI

Prima di immagazzinare informazioni in un file dovete creare un canale da usare per trasferire i dati dal computer al drive. Imparerete ad aprire e chiudere questi canali.

SCRIVERE INFORMAZIONI IN UN FILE

Imparerete a creare file nei quali immagazzinare le informazioni.

RICHIAMARE INFORMAZIONI DA UN FILE

Una volta che l'informazione è stata immagazzinata dovete essere in grado di richiamarla. Esamineremo i metodi per richiamare le informazioni da un file.

ESEMPI DI PROGRAMMI

Avrete bisogno spesso di cambiare i dati immagazzinati su file. Imparerete come si fa. Scriverete anche programmi che estraggono informazioni dai file.

12.2 Esercizi di scoperta

L'uso dei file è ormai, nei linguaggi avanzati, estremamente semplice. Vi accorgerete che scrivere programmi che fanno uso di file richiede l'utilizzo di tecniche e concetti imparati nei capitoli precedenti.

1. Accendete il computer, la TV e il drive. Mettete un disco formattato nel drive (v. Capitolo 3). Cancellate la memoria.

2. In questi esercizi non usate il ? come abbreviazione di PRINT. Battete il seguente programma:

```
90 OPEN 15,8,15
100 OPEN 2,8,2,"LISTA DONI,L," + CHR$(50)
110 PRINT "NOME: ";
120 INPUT NOME$
130 PRINT "PREZZO: ";
140 INPUT PREZ
150 PRINT#15,"P"CHR$(2)CHR$(3)CHR$(0)CHR$(1)
160 PRINT#2,NOME$;",";PREZ
170 CLOSE 2
180 CLOSE 15
190 END
```

   Un record è un insieme organizzato di informazioni. Per immagazzinare un record su dischetto dovete essere in grado di trasferire due tipi di informazione: le informazioni sulla posizione del record e quelle contenute nel record.
   La riga 90 apre un canale per i comandi di indirizzo; la riga 100 apre un canale per la trasmissione dei dati. Questi due canali insieme saranno chiamati canale di informazioni. I 15 nelle righe 90, 150 e 180 si riferiscono al canale di indirizzamento; i 2 nelle righe 100, 150, 160 e 170 si riferiscono al canale di trasmissione dati. L'8 si riferisce all'unità di gestione del disco. La "LISTA DONI,L,"+CHR$(50) crea il file LISTA DONI stabilendo la lunghezza di ciascun record a 50. Il 3 nella riga 150 indica che state per mettere informazioni nel record 3 del file.

Visualizzate il programma e controllatelo, specialmente i punti e virgola e le virgole. Notate la mancanza di spazi tra PRINT e # nelle righe 150 e 160.

3. Eseguite il programma ed inserite il vostro nome ed una cifra adeguata per il dono. Il disco comincia a "ronzare"?

   ....

   Cosa è successo alle informazioni rappresentate da NOME$ e PREZ?

   ....

4. Salvate questo programma come WRITEDISK (vedi Cap. 3).

5. Battete

```
LOAD"$",8
LIST
```

   per ottenere una lista dei file di programmi sul vostro dischetto. I programmi BASIC C-64 sono identificati sul catalogo con PRG nella terza colonna. Cosa compare nella terza colonna per il file LISTA DONI?

   ....

   REL sta per relativo, il nome tecnico del tipo di file che state usando.

6. Cancellate la memoria e battete il seguente programma.

```
90 OPEN 15,8,15
100 OPEN 2,8,2,"LISTA DONI,L," + CHR$(50)
110 PRINT#15,"P"CHR$(2)CHR$(3)CHR$(0)CHR$(1)
120 INPUT#2,NOME$,PREZ
130 CLOSE 2
140 CLOSE 15
150 END
```

   Visualizzatelo, controllatelo ed eseguitelo. Sta "ronzando" il disco?

   ....

   La riga 120 visualizza sullo schermo un punto di domanda?

   ....

Il programma ha visualizzato qualcosa sullo schermo?

....

7. Aggiungete la riga 145 come segue:

```
145 PRINT NOME$,PREZ
```

Eseguite il programma. Nella riga 110 usate l'istruzione PRINT#15 per passare l'informazione di indirizzamento ("P"). Il drive localizza il file LISTA DONI (CHR$(2)), il record 3 (CHR$(3)). L'informazione è letta dal record nella variabile NOME$ e PREZ nella riga 120. È sufficiente a rendere utile l'informazione?

....

8. Salvate questo programma come READDISK.

9. Dovete aprire e chiudere un canale ogni volta che lo usate. Per scrivere in un file usate un'istruzione PRINT# per collocare il corretto record e un'istruzione PRINT# per trasferire le informazioni nel record scelto su dischetto. Per leggere da un file usate un'istruzione PRINT# per collocare il record corretto e INPUT# per trasferire i dati da questo record alle variabili in memoria.

10. Ora passiamo ad usare file con più di un record. Caricate il programma WRITEDISK e visualizzatelo. Aggiungete le seguenti righe:

```
80 LET I=1
105 PRINT "BATTI FINE QUANDO VUOI FINIRE"
125 IF NOME$="FINE" THEN CLOSE 2:CLOSE 15:END
183 LET I=I + 1
185 GOTO 90
```

11. Cambiate la riga 150 così:

```
150 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
```

Visualizzate nuovamente il programma per controllarlo. Il programma dovrebbe ora presentarsi così:

```
80 LET I=1
90 OPEN 15,8,15
100 OPEN 2,8,2,"LISTA DONI,L," + CHR$(50)
```

```
105 PRINT "BATTI FINE QUANDO VUOI FINIRE"
110 PRINT "NOME: ";
120 INPUT NOME$
125 IF NOME$= "FINE" THEN CLOSE 2:CLOSE 15:END
130 PRINT "PREZZO: ";
140 INPUT PREZ
150 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
160 PRINT#2,NOME$;",";PREZ
170 CLOSE 2
180 CLOSE 15
183 LET I=I + 1
185 GOTO 90
190 END
```

Eseguitelo. Alla richiesta di input immettete i seguenti nomi e cifre:

ANNA 15000
ROBERTO 35000
GIACOMO 75000
SUSANNA 45000

Cosa battete per terminare l'immissione dell'informazione?

....

Provate per vedere se avete indovinato.

12. Salvate questo nuovo programma con lo stesso nome WRITEDISK battendo

```
SAVE "@0:WRITEDISK",8
```

13. Caricate READDISK. Visualizzate il programma. Cancellate le righe 130, 140, 145 e 150. Aggiungete le seguenti righe:

```
80 LET I=1
135 PRINT NOME$,PREZ
137 LET I=I + 1
139 GOTO 110
140 CLOSE 2
150 CLOSE 15
160 END
```

14. Cambiate la riga 110 così:

```
110 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
```

Il programma dovrebbe apparire come segue:

```
80 LET I=1
90 OPEN 15,8,15
100 OPEN 2,8,2,"LISTA DONI,L," + CHR$(50)
110 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
120 INPUT#2,NOME$,PREZ
135 PRINT NOME$,PREZ
137 LET I=I + 1
139 GOTO 110
140 CLOSE 2
150 CLOSE 15
160 END
```

Controllatelo per assicurarvi che la punteggiatura sia corretta. Cosa pensate che accadrà quando il programma verrà eseguito?

....

Eseguite il programma e guardate se avevate ragione.

15. Premete RUN/STOP/RESTORE. Potete evitare il lampeggiamento della spia rossa sul disco aggiungendo la riga 115 e la subroutine che segue:

```
115 GOSUB 500
500 REM SUBROUTINE DI CONTROLLO
510 INPUT#15,NUMERR
520 IF NUMERR=50 THEN 140
530 RETURN
```

Eseguite il programma. Si è accesa la luce rossa lampeggiante?

....

16. Ora ricavate alcune informazioni dal file. Visualizzate il programma. Aggiungete la seguente riga:

```
134 IF PREZ<50000 THEN 137
```

Eseguite il programma.
Sono stati visualizzati tutti i nomi?

....

17. Potete acquisire informazioni dal file. Visualizzate il programma; cambiate le righe 160 e 134 e aggiungete le righe 70 e 170 come segue:

```
160 PRINT "CI SONO ";C;"DONI SOPRA 50000 LIRE"
134 IF PREZ>50000 THEN LET C=C + 1
70 LET C=0
170 END
```

Visualizzate il programma che dovrebbe apparire così:

```
70 LET C=0
80 LET I=1
90 OPEN 15,8,15
100 OPEN 2,8,2,"LISTA DONI,L," + CHR$(50)
110 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
115 GOSUB 500
120 INPUT#2,NOME$,PREZ
134 IF PREZ>50000 THEN LET C=C + 1
135 PRINT NOME$,PREZ
137 LET I=I + 1
139 GOTO 110
140 CLOSE 2
150 CLOSE 15
160 PRINT "CI SONO ";C;"DONI SOPRA 50000 LIRE"
170 END
500 REM SUBROUTINE DI CONTROLLO
510 INPUT#15,NUMERR
520 IF NUMERR=50 THEN 140
530 RETURN
```

Una volta eseguito il programma cosa pensate che verrà visualizzato?

....

Eseguite il programma. Ciò che viene visualizzato concorda con le informazioni che avete scritto sul file?

....

18. Visualizzate il programma. Cancellate la riga 160 e cambiate le righe 70, 134 e 150 come segue:

```
70 LET SOMMA=0
134 LET SOMMA=SOMMA + PREZ
150 PRINT "IL COSTO TOTALE DEI DONI E´ ";SOMMA
```

Visualizzate il programma ed eseguitelo. Qual era il totale dei doni nelle informazioni che avete inserito?

....

Cancellate la riga 135 ed eseguite il programma. Questa volta è stata stampata l'informazione contenuta in ciascun record?

....

19. Qui finiscono gli esercizi di scoperta. Spegnete il computer e la TV; togliete il dischetto e spegnete il drive.

12.3 Analisi

APRIRE E CHIUDERE I CANALI

Quando si usa un file su dischetto si lavora con canali di informazione. Questi canali sono di due tipi: per il trasferimento di informazioni sull'indirizzamento di un record (comandi al drive) e per il trasferimento di dati nel record (o del record).
I canali sono necessari per trasferire il contenuto di record dal computer al disco e viceversa. Il canale per i comandi di indirizzamento è sempre stabilito con OPEN 15,8,15. Le istruzioni OPEN e CLOSE aprono e chiudono i canali di informazione. Per esempio le righe di programma

```
200 OPEN 15,8,15
300 OPEN 2,8,2,"FILE UNO,L," + CHR$(70)
400 CLOSE 2
500 CLOSE 15
```

aprono e chiudono i due canali di informazione al file FILE UNO. La riga 200 apre il canale di indirizzamento, mentre la riga 300 apre il canale di dati per il file FILE UNO. La riga 300 determina, se necessario, il file come file relativo (REL), il tipo di file usato negli Esercizi di scoperta. Sempre nella riga 300 ogni record in FILE UNO è stabilito a lunghezza 70 dalla L e CHR$(70). Le righe 400 e 500 chiudono il canale di informazione per FILE UNO, chiudendo ambedue i canali 2 e 15.

SCRIVERE INFORMAZIONI IN UN FILE

Una volta che un canale di informazione è aperto, potete scrivere e leggere in un file. Per scrivere in un file si usa l'istruzione PRINT# con quattro funzioni CHR$. Per esempio:

```
110 PRINT#15,"P"CHR$(2)CHR$(3)CHR$(0)CHR$(1)
```

usa il canale di indirizzamento per individuare il file associato al canale 2 di trasferimento dati e posiziona ("P") il file al record 3. CHR$(1) alla fine della riga 110 indica che desiderate accedere al record 3 dalla posizione 1. Il terzo CHR$ è usato solo se bisogna accedere a più di 255 record. Consultate la pagina 35 del manuale d'uso del drive.
Per scrivere informazioni in un file, deve essere usato un altro PRINT#. Questo PRINT# usa il canale dati 2. Ad esempio, le due righe

```
110 PRINT#15,"P"CHR$(2)CHR$(3)CHR$(0)CHR$(1)
120 PRINT#2,NOME$;",";RISUL
```

stabiliranno il valore della variabile stringa NOME$ e il valore della variabile numerica RISUL nel terzo record del file associato con il canale dati 2 che inizia ancora alla posizione 1. La riga 120 mette una virgola tra i due valori come un separatore. INPUT# usa questi separatori per stabilire dove finiscono le variabili. Nell'esempio seguente un canale di informazione è inizializzato al file QUIZ1 e permette ad alcune informazioni di essere scritte su questo file.

```
90 LET I=1
100 OPEN 15,8,15
110 OPEN 4,8,4,"QUIZ1,L," + CHR$(50)
120 PRINT "NOME: ";
130 INPUT NOME$
140 IF NOME$="FINE" THEN CLOSE 4:CLOSE 15:END
150 PRINT "RISULTATO: ";
160 INPUT RISUL$
170 PRINT#15,"P"CHR$(4)CHR$(I)CHR$(0)CHR$(1)
180 PRINT#4,NOME$;",";RISUL$
190 LET I=I + 1
200 CLOSE 4
210 CLOSE 15
220 GOTO 100
230 END
```

Nelle righe 100 e 110 è aperto il canale di informazione con QUIZ1 (il file QUIZ1 è creato se necessario). A ciascun record è assegnata una lunghezza di 50 caratteri. Nella riga 170 è individuato il record I. Nella 180, NOME$, una virgola e RISUL$ sono scritti nel record I del file associato

al canale dati 4 (QUIZ1). Le righe 200 e 210 chiudono il canale di informazione.

Usate due istruzioni PRINT# per scrivere in un file

RICHIAMARE INFORMAZIONI DA UN FILE

Per leggere informazioni da un file si usano PRINT# e INPUT# come nel programma seguente:

```
90 LET I=1
100 OPEN 15,8,15
110 OPEN 6,8,6,"QUIZ1,L," + CHR$(50)
120 PRINT#15,"P"CHR$(6)CHR$(I)CHR$(0)CHR$(1)
130 INPUT#15,NUMERR
140 IF NUMERR=50 THEN CLOSE 6:CLOSE 15:END
150 INPUT#6,NOME$,RISUL$
160 PRINT NOME$,RISUL$
170 LET I=I + 1
180 GOTO 120
190 END
```

La riga 120 è simile alla 170 del programma precedente. Essa individua il file associato con il canale dati 6 (QUIZ1) e pone quel file al record I, posizione 1. Ricordate che senza l'istruzione nella riga 160, il computer non visualizza niente.
L'istruzione INPUT# alla riga 150 legge semplicemente i dati nelle variabili: ciò che poi viene fatto con esse dipende dall'uscita che si desidera. Come nel caso precedente, PRINT# e INPUT# devono andare in coppia.

Usate le istruzioni PRINT# e INPUT# per leggere un file

Un "numero di errore" è inviato sul canale comandi (15) per ogni accesso al disco. Potete leggere il numero d'errore con l'istruzione INPUT#15. Se non ci sono errori, il numero d'errore è 0. Nella riga 130 il canale comandi trova il numero d'errore (NUMERR) dell'ultimo accesso al disco.
Nella riga 140 si stabilisce se il numero d'errore è 50 (RECORD NOT PRESENT). Se il record è presente, avete raggiunto la fine del file di record e vengono chiusi i due settori del canale di comando. Tecniche più elaborate di trattamento del file e degli errori sono contenute nel manuale d'uso del drive VIC-1541. Ci addentreremo in alcune di queste tecniche.
Sul BASIC C-64 è disponibile anche un altro tipo di file, chiamato file sequenziale. Avete visto che i file ad accesso diretto permettono la lettura e

la scrittura da e in ogni record indicando il numero del record che si vuole usare.
Leggere file sequenziali è molto simile ad usare l'istruzione READ con l'informazione contenuta in un programma nelle istruzioni DATA. Certi compiti sono eseguiti meglio con file sequenziali che con file ad accesso diretto. Informazioni sui file sequenziali sono contenute nel manuale d'uso del drive VIC-1541, al capitolo 5.

12.4 Esempi di programmi

Scriveremo una serie di programmi che possono essere utilizzati per costituire e mantenere una mailing list.

ESEMPIO 1 — MAILING LIST: PROGRAMMA DI CARICAMENTO DATI

Una mailing list contiene nomi, indirizzi e altre informazioni di persone fisiche o aziende. Ad esempio un programma di mailing list può contenere le seguenti voci:

COGNOME: (da inserire)
NOME: (da inserire)
VIA: (da inserire)
CITTÀ: (da inserire)
CAP: (da inserire)
REDDITO: (da inserire)

Il programma dovrà richiedere queste informazioni per ciascun nuovo soggetto inserito nella lista. Potete programmare il computer affinché esca dal loop di acquisizione quando viene inserita la parola FINE. Le informazioni verranno salvate in un file chiamato MAILING LIST. Alla riga 1020 viene verificata la correttezza della registrazione. Segue il programma completo.

```
100 LET I=2
110 OPEN 15,8,15
120 OPEN 2,8,2,"MAILING LIST,L," + CHR$(120)
130 PRINT "COGNOME: ";
140 INPUT COGN$
150 IF COGN$="FINE" THEN 310
160 PRINT "NOME: ";
170 INPUT NOME$
180 PRINT "VIA: ";
```

```
190 INPUT VIA$
200 PRINT "CITTA': ";
210 INPUT CITTA$
220 PRINT "CODICE POSTALE: ";
230 INPUT CAP$
240 PRINT "REDDITO: ";
250 INPUT RED
260 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
270 GOSUB 1000
280 PRINT#2,COGN$;",";NOME$;",";VIA$;
    ",";CAP$;",";RED
290 LET I=I + 1
300 GOTO 130
310 GOSUB 2000
320 CLOSE 2
330 CLOSE 15
340 END
1000 REM VERIFICA LA PRESENZA DEL RECORD
1010 INPUT#15,ERR,ERR$
1020 IF ERR=50 OR ERR=0 THEN RETURN
1030 PRINT ERR,ERR$
1040 CLOSE 2
1050 CLOSE 15
1060 END
1070 RETURN
2000 REM METTE IL NUMERO DEI RECORD E
2010 REM LA LUNGHEZZA DEL FILE NEL RECORD 1
2020 PRINT#15,"P"CHR$(2)CHR$(1)CHR$(0)CHR$(1)
2030 GOSUB 1000
2040 PRINT#2,I-1;",";120
2050 RETURN
```

Salvate il programma come ENTERDATA; lo useremo ancora.

ESEMPIO 2 — PROGRAMMA DI AGGIUNTA RECORD

Questo programma permette di aggiungere record a un dato file. Il programma precedente ENTERDATA richiede una piccola modifica. Questa modifica consiste semplicemente nella cancellazione della riga 100 e nell'aggiunta delle righe da 123 a 127 come mostrato qui sotto.

```
110 OPEN 15,8,15
120 OPEN 2,8,2,"MAILING LIST,L," + CHR$(120)
123 PRINT#15,"P"CHR$(2)CHR$(1)CHR$(0)CHR$(1)
124 GOSUB1000
125 INPUT#2,RN
127 LET I=RN + 1
130 PRINT"COGNOME: ";
140 INPUT COGN$
```

```
150 IF COGN$="FINE" THEN 310
160 PRINT "NOME: ";
170 INPUT NOME$
180 PRINT "VIA: ";
190 INPUT VIA$
200 PRINT "CITTA': ";
210 INPUT CITTA$
220 PRINT "CODICE POSTALE: ";
230 INPUT CAP$
240 PRINT "REDDITO: ";
250 INPUT RED
260 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
270 GOSUB 1000
280 PRINT#2,COGN$;",";NOME$;",";VIA$;
    ",";CAP$;",";RED
290 LET I=I + 1
300 GOTO 130
310 GOSUB 2000
320 CLOSE 2
330 CLOSE 15
340 END
1000 REM VERIFICA LA PRESENZA DEL RECORD
1010 INPUT#15,ERR,ERR$
1020 IF ERR=50 OR ERR=0 THEN RETURN
1030 PRINT ERR,ERR$
1040 CLOSE 2
1050 CLOSE 15
1060 END
1070 RETURN
2000 REM METTE IL NUMERO DEI RECORD E
2010 REM LA LUNGHEZZA DEL FILE NEL RECORD 1
2020 PRINT#15,"P"CHR$(2)CHR$(1)CHR$(0)CHR$(1)
2030 GOSUB 1000
2040 PRINT#2,I-1;",";120
2050 RETURN
```

Salvate questo programma chiamandolo ADDRECORD.

ESEMPIO 3 — PROGRAMMA DI STAMPA DI ETICHETTE

Questo programma usa il file relativo MAILING LIST per produrre la stampa di etichette per buste. Le etichette dovrebbero avere tre righe come mostrato nel seguente esempio:

```
GATTI PAOLO
VIA NEWTON 3
20126 MILANO
```

Il programma completo è:

```
100 LET I=2
110 OPEN 15,8,15
120 OPEN 2,8,2,"MAILING LIST,L," + CHR$(120)
125 REM LEGGE IL NUMERO DELL'ULTIMO RECORD
130 PRINT#15,"P"CHR$(2)CHR$(1)CHR$(0)CHR$(1)
140 INPUT#2,RN
145 REM LEGGE IL RECORD
150 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
160 INPUT#2,COGN$,NOME$,VIA$,CITTA$,CAP$,RED
165 REM STAMPA L'ETICHETTA
170 PRINT COGN$;",";NOME$
180 PRINT VIA$
190 PRINT CAP$;" ";CITTA$
200 PRINT
210 PRINT
220 LET I=I + 1
230 IF I <= RN THEN 150
240 CLOSE 2
250 CLOSE 15
260 END
```

Salvate il programma con il nome LABEL.

ESEMPIO 4 — PROGRAMMA DI SELEZIONE DI ETICHETTE

Avendo scritto il precedente programma, potete facilmente selezionare record in un file in base a specifiche caratteristiche. In questo esempio potrete ottenere una serie di indirizzi di tutti quei nominativi il cui reddito supera i 20 milioni. Modificate il precedente programma cambiando semplicemente la riga 165 e aggiungendo la 167. Ecco il programma completo.

```
100 LET I=2
110 OPEN 15,8,15
120 OPEN 2,8,2,"MAILING LIST,L," + CHR$(120)
125 REM LEGGE IL NUMERO DELL'ULTIMO RECORD
130 PRINT#15,"P"CHR$(2)CHR$(1)CHR$(0)CHR$(1)
140 INPUT#2,RN
145 REM LEGGE IL RECORD
150 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
160 INPUT#2,COGN$,NOME$,VIA$,CITTA$,CAP$,RED
165 REM SELEZIONA GLI INDIRIZZI
166 REM CON REDDITO <= 20 MILIONI
167 IF RED <=20E6  THEN 220
170 PRINT COGN$;",";NOME$
```

```
4030 IF ANS$="Y" THEN 150
4040 GOTO 430
4050 RETURN
```

Salvate questo programma come MODIFYRECORD.

ESEMPIO 6 — UNA COMPLETA MAILING LIST

ADDRECORD, LABEL, SELECTEDLABEL e MODIFYRECORD possono essere riuniti come subroutine di un programma principale. In questo modo sarà possibile scegliere una o più funzioni su un menu visualizzato da un semplice programma. Potrete usare questo programma per gestire una mailing list personale. Questo programma dovrebbe iniziare visualizzando un menu con tutte le opzioni disponibili.

```
0.    REGISTRAZIONE INIZIALE DATI
1.    AGGIUNTA DATI
2.    STAMPA TOTALE DI INDIRIZZI
3.    STAMPA SELEZIONATA DI INDIRIZZI
4.    MODIFICA DATI
5.    FINE
```

Segue il listato.

```
100 PRINT "0.    REGISTRAZIONE INIZIALE DATI"
110 PRINT "1.    AGGIUNTA DATI"
120 PRINT "2.    STAMPA TOTALE DI INDIRIZZI"
130 PRINT "3.    STAMPA SELEZIONATA DI INDIRIZZI"
140 PRINT "4.    MODIFICA DATI"
150 PRINT "5.    FINE
155 PRINT
160 PRINT "SCEGLIETE UN NUMERO ";
170 INPUT N
180 IF N=5 THEN END
190 IF N=0 THEN 10000
200 ON N GOTO 11000,12000,13000,14000
```

Ciascuna delle cinque subroutine dovrebbe essere rinumerata per essere incorporata nel programma principale; andranno inoltre aggiunte delle REM all'inizio di ciascuna di esse. Per far tornare il controllo al programma principale dopo l'esecuzione delle subroutine (che iniziano alle righe 10000, 11000, 13000, 14000) dovremo sostituire l'istruzione END alla fine di ciascuna di esse con un GOTO 110. Per esempio ADDRECORD dovrà iniziare con:

```
11000 REM SUBROUTINE ADDRECORD
11110 OPEN 15,8,15
```

mentre l'END alla riga 340 deve diventare:

```
11340 GOTO 110
```

Le routine rimandano alla 110 poiché ENTERDATA deve girare solo la prima volta.

ESEMPIO 7 — ACCESSO AI RECORD

Negli esempi precedenti avete usato i comandi PRINT# e INPUT# per scrivere e leggere un record in un file con le virgole come separatori di dati. In questo modo non è stato necessario curare l'esatta posizione di ciascun elemento in un record. In questo esempio invece potremo posizionare ciascun elemento (o campo) di un record in una posizione specifica, attraverso l'utilizzo dell'ultima funzione CHR$ nell'istruzione "P" PRINT#. Sono necessari ulteriori controlli della lunghezza delle stringhe per evitare che un campo si sovrapponga al successivo. Nell'esempio il record è composto di tre campi: due stringhe e uno numerico. I campi vengono letti dal record memorizzato su dischetto partendo dalla seconda posizione di ciascuno di essi. Segue il listato del programma.

```
100 OPEN 15,8,15
110 OPEN 2,8,2,"ESEMPIO,L," + CHR$(40)
120 READ DES$,PREZZO,COLR$
130 DATA CAPPELLO,27000,ROSA
140 REM
150 LET SPAZI$="                              "
160 LET DES$=MID$(DES$,1,14)
170 LET PAD=14 - LEN(DES$)
180 LET DES$=DES$ + MID$(SPAZI$,1,PAD)
190 LET COLR$=MID$(COLR$,1,PAD)
200 LET PAD=7 - LEN(COLR$)
210 LET COLR$=COLR$+MID$(SPAZI$,1,PAD)
220 REM SCRIVE NELLE POSIZIONI 1,16,32
230 LET P=1
240 GOSUB 2000
250 PRINT#2,DES$
260 LET P=16
270 GOSUB 2000
280 PRINT#2,PREZZO
290 LET P=32
300 GOSUB 2000
310 PRINT#2,COLR$
```

```
4030 IF ANS$="Y" THEN 150
4040 GOTO 430
4050 RETURN
```

Salvate questo programma come MODIFYRECORD.

ESEMPIO 6 — UNA COMPLETA MAILING LIST

ADDRECORD, LABEL, SELECTEDLABEL e MODIFYRECORD possono essere riuniti come subroutine di un programma principale. In questo modo sarà possibile scegliere una o più funzioni su un menu visualizzato da un semplice programma. Potrete usare questo programma per gestire una mailing list personale. Questo programma dovrebbe iniziare visualizzando un menu con tutte le opzioni disponibili.

```
0.   REGISTRAZIONE INIZIALE DATI
1.   AGGIUNTA DATI
2.   STAMPA TOTALE DI INDIRIZZI
3.   STAMPA SELEZIONATA DI INDIRIZZI
4.   MODIFICA DATI
5.   FINE
```

Segue il listato.

```
100 PRINT "0.   REGISTRAZIONE INIZIALE DATI"
110 PRINT "1.   AGGIUNTA DATI"
120 PRINT "2.   STAMPA TOTALE DI INDIRIZZI"
130 PRINT "3.   STAMPA SELEZIONATA DI INDIRIZZI"
140 PRINT "4.   MODIFICA DATI"
150 PRINT "5.   FINE
155 PRINT
160 PRINT "SCEGLIETE UN NUMERO ";
170 INPUT N
180 IF N=5 THEN END
190 IF N=0 THEN 10000
200 ON N GOTO 11000,12000,13000,14000
```

Ciascuna delle cinque subroutine dovrebbe essere rinumerata per essere incorporata nel programma principale; andranno inoltre aggiunte delle REM all'inizio di ciascuna di esse. Per far tornare il controllo al programma principale dopo l'esecuzione delle subroutine (che iniziano alle righe 10000, 11000, 13000, 14000) dovremo sostituire l'istruzione END alla fine di ciascuna di esse con un GOTO 110. Per esempio ADDRECORD dovrà iniziare con:

```
11000 REM SUBROUTINE ADDRECORD
11110 OPEN 15,8,15
```

mentre l'END alla riga 340 deve diventare:

```
11340 GOTO 110
```

Le routine rimandano alla 110 poiché ENTERDATA deve girare solo la prima volta.

ESEMPIO 7 — ACCESSO AI RECORD

Negli esempi precedenti avete usato i comandi PRINT# e INPUT# per scrivere e leggere un record in un file con le virgole come separatori di dati. In questo modo non è stato necessario curare l'esatta posizione di ciascun elemento in un record. In questo esempio invece potremo posizionare ciascun elemento (o campo) di un record in una posizione specifica, attraverso l'utilizzo dell'ultima funzione CHR$ nell'istruzione "P" PRINT#. Sono necessari ulteriori controlli della lunghezza delle stringhe per evitare che un campo si sovrapponga al successivo. Nell'esempio il record è composto di tre campi: due stringhe e uno numerico. I campi vengono letti dal record memorizzato su dischetto partendo dalla seconda posizione di ciascuno di essi. Segue il listato del programma.

```
100 OPEN 15,8,15
110 OPEN 2,8,2,"ESEMPIO,L," + CHR$(40)
120 READ DES$,PREZZO,COLR$
130 DATA CAPPELLO,27000,ROSA
140 REM
150 LET SPAZI$="                              "
160 LET DES$=MID$(DES$,1,14)
170 LET PAD=14 - LEN(DES$)
180 LET DES$=DES$ + MID$(SPAZI$,1,PAD)
190 LET COLR$=MID$(COLR$,1,PAD)
200 LET PAD=7 - LEN(COLR$)
210 LET COLR$=COLR$+MID$(SPAZI$,1,PAD)
220 REM SCRIVE NELLE POSIZIONI 1,16,32
230 LET P=1
240 GOSUB 2000
250 PRINT#2,DES$
260 LET P=16
270 GOSUB 2000
280 PRINT#2,PREZZO
290 LET P=32
300 GOSUB 2000
310 PRINT#2,COLR$
```

```
320 PRINT "DATI SCRITTI SUL FILE"
330 PRINT DES$,PREZZO,COLR$
340 REM LEGGE I DATI DAL FILE
350 LET P=1
360 GOSUB 2000
370 INPUT#2,FDES$
380 LET P=16
390 GOSUB 2000
400 INPUT#2,FPREZZO
410 LET P=32
420 GOSUB 2000
430 INPUT#2,FCOLR$
440 PRINT
450 PRINT "DATI RILETTI DAL FILE"
460 PRINT FDES$,FPREZZO,FCOLR$
470 CLOSE 2
480 CLOSE 15
490 END
2000 REM METTE P NEL RECORD 15
2010 PRINT#15,"P"CHR$(2)CHR$(5)CHR$(0)CHR$(P)
2020 RETURN
```

12.5 Problemi

1. Disegnate la struttura appropriata di un record per un file ad accesso diretto usato per indicizzare la vostra collezione di cassette. Indicate la lunghezza massima di ciascun elemento nel record e la lunghezza totale del record. Stabilite per ciascun elemento della struttura del record nomi di variabili valide per il BASIC C-64.

2. Disegnate la struttura appropriata di un record per un file ad accesso casuale da usare per seguire l'andamento del vostro conto in banca.

3. Disegnate la struttura appropriata di un record per un file ad accesso casuale da usare per eseguire l'inventario della vostra dispensa.

4. Scrivete la struttura appropriata di un record per un file da usare per gestire la mailing list. Ricordatevi che compleanni e anniversari sono importanti per la vostra corrispondenza personale.

5. Scrivete un programma che usi un file per gestire i vostri acquisti. Ciascun record dovrebbe avere la seguente struttura:

Variabile	Descrizione	Lunghezza approssimativa
CART$	Carta di credito	20
NOME$	Nome del negozio	30
DATE$	Data dell'acquisto	10
DES$	Descrizione dell'acquisto	50
IMP	Importo speso	8

Il programma dovrebbe permettervi di calcolare il totale degli importi addebitati su ciascuna carta di credito.

6. Scrivete un programma che usi la struttura del record del problema 4 per gestire la vostra mailing list personale. Il programma dovrebbe permettervi di stampare gli indirizzi per gli auguri di Natale e per i messaggi agli amici.

12.6 Test di apprendimento

1. Se viene eseguito il seguente programma:

```
90 OPEN 15,8,15
100 OPEN 2,8,2,"FILE DUE,L," + CHR$(70)
110 PRINT#15,"P"CHR$(2)CHR$(5)CHR$(0)CHR$(1)
120 PRINT#2,"CIAO";",";"MARCO"
130 CLOSE 2
140 CLOSE 15
150 END
```

a. Quale file sarà usato?

....

b. Quanti caratteri sono permessi in ciascun record?

....

c. Quanti elementi sono inseriti nel record?

....

d. In quale record verrà scritto?

....

2. Scrivete una riga di programma che apra un file relativo chiamato INVENTARIO con un record di lunghezza 45.

....

3. Scrivete un programma che legga cinque elementi dal record 75 in un file relativo chiamato TEST1 con un record di lunghezza 50. Assicuratevi di chiudere il file.

....

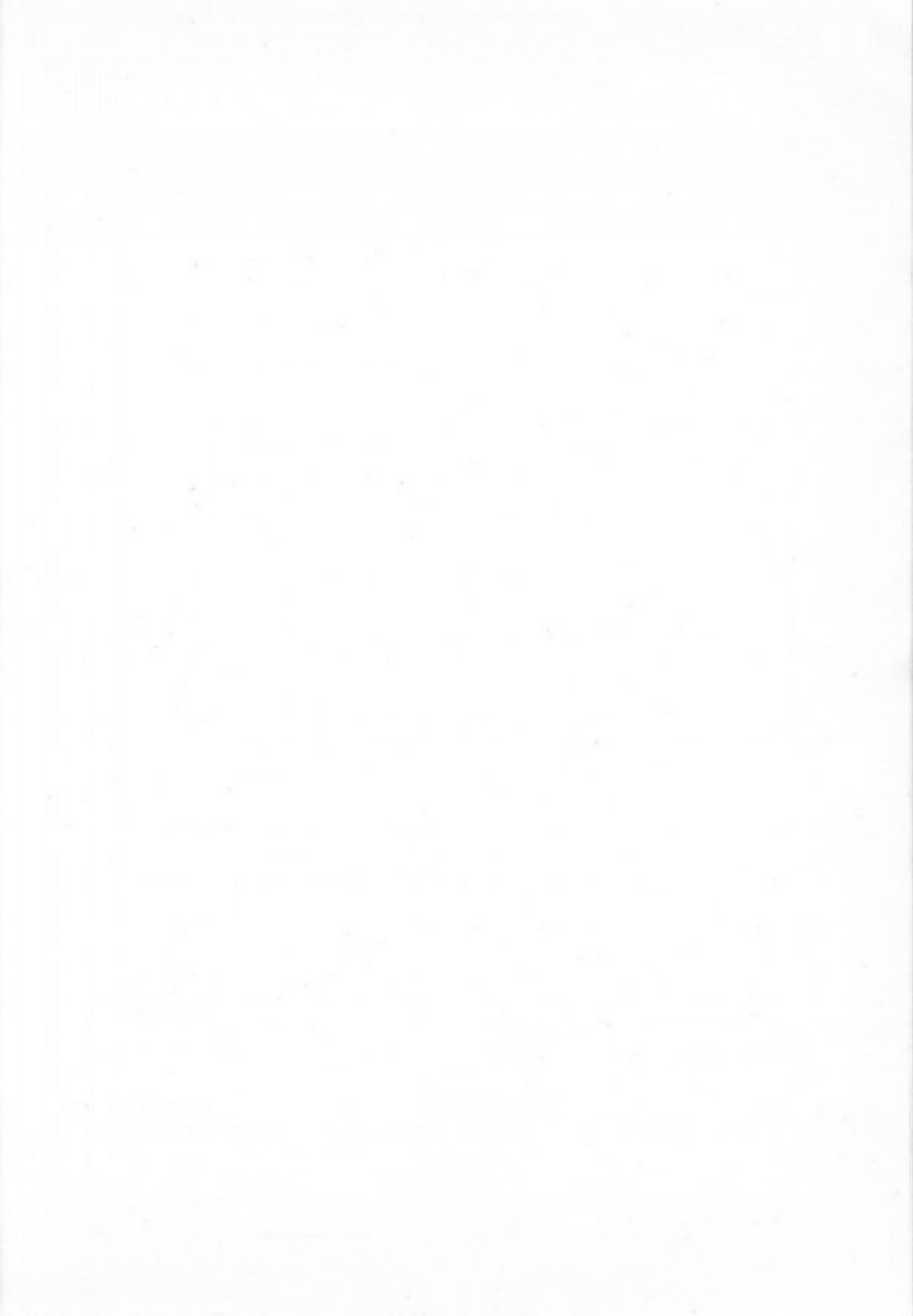
4. Cosa c'è di sbagliato nella seguente riga di programma?

```
200 OPEN 2,8,2,"FILE UNO,L", CHR$(70)
```

....

5. Cosa c'è di sbagliato nelle seguenti righe di programma che dovrebbero leggere COGN$ dal record 5 di un file chiamato FILE UNO?

```
200 OPEN 15,8,15
210 OPEN 2,8,2,"FILE UNO,L," + 100
220 PRINT# 15,CHR$(5)CHR$(2)CHR$(0)CHR$(1)
230 INPUT COGN$
```

Appendice

Programma di supporto DOS

IL C-64 WEDGE

Il C-64 Wedge o programma di supporto DOS sul disco Demo (Test/Demo) permette di controllare il drive più facilmente. Potrebbe essere interessante per voi copiare questo programma e il DOS 5.1 su ciascuno dei dischetti che usate. Per utilizzarlo dovete battere

```
LOAD "C-64 WEDGE",8
RUN
```

ogni volta che accendete il computer. Una volta che il programma è caricato ed eseguito, potete usare la barra (/) per caricare i programmi senza bisogno delle virgolette, della virgola e del numero 8.
Ad esempio:

```
/MEDIA
```

caricherà il programma MEDIA in memoria. Il tasto @ viene usato per mandare comandi al drive. Ad esempio:

```
@$
```

visualizza il catalogo e i file sullo schermo senza cancellare il programma in memoria. @ è usata per formattare i dischetti e per cancellare programmi e file da dischetto. Ad esempio:

```
@NEW0:PROVA,01
```

formatta un dischetto come PROVA con ID 01, e

```
@SCRATCH0:CIAO
```

cancella il programma CIAO dal dischetto nel drive. Infine, gli errori del drive segnalati da una luce rossa lampeggiante, possono essere letti dal drive e visualizzati sullo schermo semplicemente battendo

```
@
```

Ulteriori informazioni sul C-64 Wedge sono contenute nel capitolo 3 del manuale d'uso del drive VIC-1541.

Appendice **B**

Guida rapida C-64

COMANDI

Cancellazione di un file Vedere il comando OPEN

CONT Fa ripartire l'esecuzione dopo l'uso del tasto RUN/STOP.

LIST Visualizza il programma in memoria
LIST Visualizza l'intero programma
LIST 40–130 Visualizza le righe dalla 40 alla 130
LIST –400 Visualizza tutte le righe fino alla 400
LIST 400– Visualizza tutte le righe dalla 400 alla fine del programma

LOAD "*nome del file*",8 Trasferisce il programma nella memoria
LOAD"PIPPO",8

NEW Cancella il programma in memoria

OPEN 15,8,15,"NEW0:*nome del disco,ID*" Formatta il dischetto con il nome del disco e l'ID; cancella ogni informazione precedente nel dischetto.
OPEN 15,8,15,"NEW0:TASSE,ZZ"

OPEN 15,8,15,"SCRATCH0:*nome del file*" Cancella il file dal dischetto
OPEN 15,8,15,"SCRATCH0:
TASSEGIUGNO"

RUN Esegue il programma in memoria

SAVE "*nome del file*",8 Scrive il programma in memoria nel dischetto e gli dà il nome specificato
SAVE"ETICHETTE",8

SAVE "@0: *nome del file*",8 Elimina il contenuto del vecchio file e lo sostituisce con il programma in memoria
SAVE"@0: ETICHETTE",8

FUNZIONI

ABS(*numero*) Calcola il valore assoluto
LET X = ABS(–7)

ASC(*stringa*) Riporta il codice ASCII del primo carattere della stringa specificata
LET A = ASC(B$)

CHR$(*numero*) Prende il numero di codice ASCII e riporta il carattere corrispondente
PRINT CHR$(65)

INT(*numero*) Riporta l'intero più grande minore o uguale del numero specificato
LET Y = INT(2.7)

LEN(*stringa*) Riporta la lunghezza della stringa
FOR X = 1 TO LEN(A$)

MID$(*stringa, n1, n2*) Riporta una stringa lunga *n2* caratteri iniziante con il n1-esimo carattere della stringa
IF MID$(A$7,4) = "NOME" THEN 300

RND(*n*) Riporta un numero casuale tra 0 (incluso) e 1 (escluso); un numero negativo rimette il seme nel generatore di numeri casuali in modo tale che successivi richiami alla RND generano la stessa sequenza di numeri casuali
LET D = 1+RND(1)*9
IF RND(−2)>.5 THEN 220

SGN(*numero*) Riporta il segno della espressione numerica specificata
LET P = SGN(R−E)

SQR(*numero*) Riporta la rappresentazione di stringa di un numero
LET N$ = STR$(27)

TAB(*n*) Non è propriamente una funzione, ma è usata con l'istruzione PRINT; sposta il cursore alla posizione specificata, sulla linea orizzontale.
PRINT "CIAO";TAB(12);N$

VAL(*stringa*) Converte una stringa di cifre in un numero
LET P = VAL(A$)

ISTRUZIONI

CLOSE *n* Chiude il canale aperto specificato da *n*
CLOSE 1

DATA Lista di dati che contiene i valori da assegnare alle variabili nell'istruzione READ
DATA 7.2, PIPPO, 3,4,5

DEF FN*v*1(*v*2) Definisce una funzione, chiamata FN*v*1, della variabile *v*2
DEF FNC(X) = Z↑2−5

DIM *v*(*n*) Riserva spazio per l'array, con le dimensioni specificate da *n*1, *n*2
DIM A(10,20),B$(50)

END Ferma l'esecuzione del programma

FOR...TO/NEXT Genera un loop
FOR J = 1 TO 10
NEXT J
FOR A = 2*X TO Y STEP 2
NEXT A

GET *v$* Assegna l'ultimo tasto premuto a *v$*
GET A$

GOSUB *num.riga* Chiama la subroutine iniziante al numero di riga specificato
GOSUB 500

GOTO *num. riga* Salta al numero di riga specificato (salto incondizionato)
GOTO 412

IF *condizione* **THEN** *num. riga*
Salta al numero di riga specificato dopo THEN quando la condizione è vera (salto condizionato)
IF X>10 THEN 320

IF *condizione* **THEN** *istruzione*
Esegue l'istruzione dopo THEN quando la condizione è vera
IF Y−2 = 7 THEN END

INPUT *v* Provoca una pausa del programma per l'input dalla tastiera e assegna il dato inserito alle variabili specificate
INPUT A,B INPUT D$

INPUT# *n,v* Fa leggere le informazioni dall'unità collegata al canale *n* e le assegna alle variabili specificate
INPUT#1,A,N$

LET *v=espressione* Assegna un valore alla variabile *v*
LET X = 7 LET C = C+1

ON *v* **GOSUB** *n1,n2* Provoca un salto ad una delle subroutine, a seconda del valore della variabile *v*
ON N GOSUB 300,350,400

OPEN 15,8,15 Apre un canale di trasmissione di comandi; è richiesto prima di altre operazioni sul file

OPEN *n*1, **8**, *n*1, "*nome del file, L,*" +**CHR$**(*n*2) Apre il canale dati *n*1 per le operazioni di input e output, lo assegna al nome del file specificato e stabilisce a *n*2 la lunghezza del record
OPEN 2,8,2, "DATI,L,"+CHR$(50)

PRINT Manda l'output all'unità specificata: in mancanza di specificazioni, è lo schermo
PRINT A,B PRINT C$,X(J)
PRINT#1,A$

READ *lista di variabili* Legge l'elemento successivo nella lista di DATA e lo assegna alla variabile specificata
READ B$,NUMERO,M(K)

REM Permette l'inserimento di un commento in una riga di programma
REM SALTA ALLA SUBROUTINE

RESTORE Ripristina i dati nelle istruzioni DATA in modo che possono essere letti nuovamente

RETURN Riporta il controllo del programma dalla subroutine all'istruzione che segue il GOSUB

TASTI SPECIALI

Controllo cursore

Tasto	Funzione
←CRSR→	Sposta il cursore a destra di uno spazio
↑CRSR↓	Sposta il cursore in giù di uno spazio
SHIFT ←CRSR→	Sposta il cursore a sinistra di uno spazio
SHIFT ↑CRSR↓	Sposta il cursore in su di uno spazio
INST/DEL	Cancella il carattere alla sinistra del cursore

SHIFT | INST/DEL

Inserisce uno spazio alla sinistra del cursore

RETURN — Segnala la fine della riga battuta

Altri tasti speciali

CLR/HOME — Sposta il cursore alla posizione HOME nello angolo in alto a sinistra dello schermo

SHIFT | CLR/HOME

Pulisce lo schermo e pone il cursore nell'angolo in alto a sinistra (HOME)

CTRL | Ø   CTRL | 9

Cambia il colore o il modo dei caratteri come indicato nella faccia anteriore dei tasti

SHIFT | C= — Permette il passaggio tra il modo maiuscolo/grafico e quello minuscolo/maiuscolo

SHIFT — Fa accedere ai caratteri grafici segnati sulla destra della faccia anteriore dei tasti, quando si è nel modo maiuscolo/grafico

C= — Fa accedere ai caratteri grafici segnati sulla sinistra della faccia anteriore dei tasti, in modo maiuscolo/grafico

RUN/STOP — Interrompe il programma

RUN/STOP | RESTORE

Blocca ogni attività, cancella lo schermo e riporta il controllo alla tastiera

Caratteri grafici

Vi si accede con i tasti SHIFT e C= nel modo maiuscolo/grafico. Sono segnati sulle facce anteriori dei tasti.

Appendice

Soluzioni dei test di apprendimento

CAPITOLO 1

1. Premere il tasto RETURN.
2. Premete RUN/STOP e RESTORE, oppure spegnete e riaccendete il computer.
3. *
4. Premere SHIFT e CLR/HOME.
5. Divisione.
6. Sullo schermo sarà visualizzato il numero 2.
7. Sullo schermo sarà visualizzata la stringa 25/5+2.
8. Spostate il cursore sulla G di PRING con il tasto DELETE che si trova sul lato destro della tastiera. Poi battete T2+3*4.

CAPITOLO 2

1. Premere il tasto RETURN.
2. Premere RUN/STOP/RESTORE.

3. Premere RUN/STOP.

4. L'istruzione PRINT C non ha numero di riga.

5. Sullo schermo sarebbe visualizzato il numero 2.

6. Fino a 77 caratteri.

7. Battere la riga usando un numero di riga non ancora presente nel programma.

8. Ribattere la riga compreso il numero di riga.

9. Battere il numero di riga e premere il tasto RETURN.

10. Battere LIST e premere RETURN.

11. Premere SHIFT/CLR/HOME.

12. Battere RUN e premere RETURN.

13. Battere NEW e premere RETURN.

14. Una variabile a stringa di caratteri termina sempre con $.

15. Visualizza la riga 120 pronta per modifiche.

16. Il tasto movimento cursore ⇐CRSR⇒ e ⇑CRSR⇓, qualche volta con SHIFT (per sinistra e alto).

17. SHIFT/INST/DEL e INST/DEL.

18. Premendo RETURN.

CAPITOLO 3

1. a) * b) ↑ c) /

2. a) Elevazione a potenza b) Moltiplicazione e divisione c) Addizione e sottrazione

3. Addizione

4. Da sinistra a destra.

5. `100 LET A=(4+3*B/D)↑2`

6. 4

7. a) 5.16E+09 b) 3.14E-05

8. a) 7258000 b) 0.001437

9. / poi + poi ↑

10. a) Battere LOAD "*nome del programma*", 8 e premere RETURN
    b) Battere SAVE "*nome del programma*", 8 e premere RETURN
    c) OPEN 15,8,15, "SCRATCH0: (*nome del programma*)"
    d) Battere NEW e premere RETURN.
    e) Battere LIST e premere RETURN.
    f) Battere RUN e premere RETURN.
    g) Battere LOAD"$", 8 RETURN e poi battere LIST RETURN.

11. Usate i tasti di movimento del cursore sulla destra della tastiera, per posizionare il cursore sull'errore. Battete il carattere giusto e premete RETURN per immettere la riga.

CAPITOLO 4

1.
```
1  2  3  4  5  6  7  8  9  10 11 12
13 14 15 16 17 18 19 20 21 22 23 24
```
   (ecc.)

2. a) Per mezzo di un'assegnamento (per esempio: 100 LET A = 3)
   b) Istruzioni INPUT c) Istruzioni READ e DATA

3. Una stringa.

4. Fornire informazioni all'interno del programma a beneficio del programmatore o dell'utente.

5. DATA.

6. Sarà visualizzato Y = 3.

7. Quattro.

8. Tante quante sono necessarie.

9. Fornire un metodo per ottenere una spaziatura variabile in uscita.

10.
```
1        3
1 3
```

11.
```
?10,12,13
?EXTRA IGNORED
22
READY
```

12.
```
100 PRINT "IMMETTERE NUM.MIGLIA";
110 INPUT N
120 LET K=1.609*N
130 PRINT N;" MIGLIA EQUIVALGONO A ";K;"CHILOMETRI"
140 END
```

CAPITOLO 5

1.
```
6
10
14
18
```

2.
```
OTTIMO


MIGLIORE
OTTIMO


BUONO
MIGLIORE
OTTIMO


?OUT OF DATA  ERROR IN 100
```

3.
```
100 PRINT "QUANTE BOTTIGLIE";
110 INPUT N
120 IF N <= 20 THEN 160
130 IF N <= 50 THEN 180
140 LET U=1500
150 GOTO 190
160 LET U=2000
170 GOTO 190
180 LET U=1800
190 LET P=N*U
200 PRINT "IL PREZZO PER BOTTIGLIA E' ";U
210 PRINT "IL COSTO TOTALE DELL'ORDINE E'";P
220 PRINT
230 GOTO 100
240 END
```

4.
```
100 LET X=0
110 PRINT X;
120 LET X=X+5
130 IF X <= 175 THEN 110
140 END
```

5.
```
100 PRINT "QUAL E'IL LIMITE DI VELOCITA' "
110 INPUT A
120 PRINT "QUAL E' LA VELOCITA' RILEVATA "
130 INPUT B
140 LET X=B-A
150 IF X <= 10 THEN 210
160 IF X <= 20 THEN 230
170 IF X <= 30 THEN 250
180 IF X <= 40 THEN 270
190 LET F=80000
200 GOTO 280
210 LET F=5000
220 GOTO 280
230 LET F=10000
240 GOTO 280
250 LET F=20000
260 GOTO 280
270 LET F=40000
280 PRINT "LA MULTA E' ";F;" LIRE"
290 END
```

CAPITOLO 6

1. `20 18 16 14 12 10 8  6  4  2`

2. `1 2 3 2 4 6 3 6 9 4 8 12`

3. a) 6 b) 7 c) 22.8 d) -1

4. I loop sono incrociati.

5.
```
100 PRINT "MIGLIA","CHILOMETRI"
110 PRINT "------","----------"
120 PRINT
130 FOR M=10 TO 100 STEP 10
140 PRINT M,1.609*M
150 NEXT M
160 END
```

6.
```
100 DATA 10
110 DATA 25,21,24,21,26,27,25,24,23,24
120 READ N
130 LET S=0
140 FOR I=1 TO N
150 READ X
160 LET S=S+X
170 NEXT I
180 PRINT S/N
190 END
```

CAPITOLO 7

1. Riservare spazio per un array.

2. X(3,4)

3.
```
100 DIM A(50)
110 PRINT "QUANTI NUMERI ";
120 INPUT N
130 PRINT "QUALI SONO I NUMERI"
140 FOR I=1 TO N
150 INPUT A(I)
160 NEXT I
170 LET S=0
180 FOR I=1 TO N
190 IF A(I) <= 0 THEN 210
200 LET S=S+A(I)
210 NEXT I
220 PRINT "LA SOMMA DEGLI ELEMENTI POSITIVI E' ";S
230 END
```

4.
```
100 FOR R=1 TO 4
110 FOR C=1 TO 6
120 LET X(R,C)=4
130 NEXT C
140 NEXT R
150 END
```

5.
```
2 0 0 0 0
0 2 0 0 0
0 0 2 0 0
0 0 0 2 0
0 0 0 0 2
```

6. a) 100 DIMA(2,3) b) A(2,3) = 4 c) A(X,Y) = A(1,2) = 3 d) A(A(1,1), A(2,2)) = A(1,2) = 3

CAPITOLO 8

1. Aggiungendo $ al nome di una variabile numerica.

2. Falso.

3. `MID$(A$,10,8)`

4.
```
100 INPUT A$
110 FOR X=LEN(A$) TO 1 STEP -1
120 PRINT MID$(A$,1,X)
130 NEXT X
140 END
```

5. A
AB
ABC
ABCD
ABCDE

(ecc.)

ABCDEFGHIJKLMNOPQRSTUVWXYZ

CAPITOLO 9

1. a) 4 b) 14 c) 30 d) 80

2.
```
2 1 3
4
```

3. a) Battere GOSUB e numero di riga all'inizio della subroutine.
b) RETURN

4. ```
   BIANCO
   ROSSO
   BLU
   ```

CAPITOLO 10

1. 6

2. Lo schermo verrà pulito e il cursore posto nella posizione HOME

3. Verrà stampata una J inversa.

4. Verrà stampato un quadrato rosso.

5. Premete il tasto C= e il tasto Q.

CAPITOLO 11

1. ```
   100 FOR I=1 TO 100
   110 LET X=INT(4*RND(1)+1)
   120 PRINT X;
   130 NEXT I
   140 END
   ```

2. ```
   100 FOR I=1 TO 100
   110 LET X=25+25*RND(1)
   120 PRINT X,
   130 NEXT I
   140 END
   ```

3. L'uscita sarà scelta a caso fra BIANCO e ROSSO. Vengono qui presentate tre uscite del programma per indicare la natura casuale del processo.

   ```
   (1)        (2)        (3)
   ROSSO      BIANCO     BIANCO
   ROSSO      BIANCO     ROSSO
   BIANCO     ROSSO      BIANCO
   BIANCO     BIANCO     BIANCO
   ROSSO      BIANCO     BIANCO
   ROSSO      ROSSO      BIANCO
   ROSSO      ROSSO      ROSSO
   BIANCO     ROSSO      BIANCO
   ROSSO      BIANCO     BIANCO
   ROSSO      BIANCO     ROSSO
   ```
   ```

4. Cinque numeri casuali nella forma X.XX nell'intervallo da 0.00 a 9.99. Sono presentate qui sotto tre uscite del programma per illustrare la natura casuale del processo.

```
 (1)      (2)      (3)
0.51     6.69     1.15
9.34     4.04     8.87
9.08     9.06     9.26
9.26     6.71     2.59
5.98     8.15     3.05
```

CAPITOLO 12

1. a) FILE DUE b) 70 c) 2 d) 5

2.
```
100 OPEN 2,8,2,"INVENTARIO,L" + CHR$(45)
```

3.
```
90 OPEN 15,8,15
100 OPEN 2,8,2,"TEST1,L" + CHR$(50)
110 PRINT#15,"P"CHR$(2)CHR$(75)CHR$(0)CHR$(1)
120 INPUT#2 A$,B$,C$,D$,E$
130 CLOSE 2
140 CLOSE 15
150 END
```

4. La riga corretta è:

```
200 OPEN 2,8,2,"FILE UNO,L" + CHR$(70)
```

5. Le righe 210 e 220 dovrebbero essere

```
210 OPEN 2,8,2,"FILE UNO,L," + CHR$(100)
220 PRINT#15,"P"CHR$(2)CHR$(5)CHR$(0)CHR$(1)
```

Appendice

Soluzioni dei problemi

CAPITOLO 4

1.
```
100 REM CAP 4,PROB 1
110 READ A,B,C,D
120 DATA 10,9,1,2
130 LET S=A+B
140 LET P=C*D
150 PRINT S,P
160 END
```

3.
```
100 REM CAP 4, PROB 3
110 READ A,B,C,D
120 DATA 21,18,6,3
130 PRINT A
140 PRINT B
150 PRINT C
160 PRINT D
170 END
```

5. Non è stato assegnato alcun valore a C.

7.
```
100 REM CAP 4,PROB 7
110 PRINT "CONTANTE = ";
120 INPUT C
130 PRINT "TITOLI COMMERCIABILI = ";
140 INPUT M
150 PRINT "CREDITI = ";
160 INPUT R
170 PRINT "DEBITI = ";
```

```
180 INPUT L
190 LET A=(C+M+R)/L
200 PRINT "INDICE = ";A
210 END
```

9. Il programma ritorna indietro alla riga 100 dove A è reso uguale a 1 dopo ciascuna uscita. Il programma si può correggere cambiando la riga 130 come segue

```
130 GOTO 110
```

11. Il punto errato si trova nelle istruzioni 100, 110 e 120. I valori di L, P e A dovrebbero essere stampati, ma non sono stati definiti. Il computer assegnerà il valore zero alle tre variabili e questi zero sono visualizzati dalle righe 100, 110 e 120. Il programma può essere corretto cancellando la riga 130 e L, P e A alla fine delle righe 100, 110 e 120. Ora è necessario aggiungere le righe seguenti.

```
105 INPUT L
115 INPUT P
125 INPUT A
```

13.
```
100 REM CAP 4,PROB 13
110 DATA 21423,21493,5
120 DATA 5270,5504,13
130 DATA 65214,65559,11.5
140 READ R1,R2,G
150 LET M=(R2-R1)/G
160 PRINT M
170 GOTO 140
180 END
```

15.
```
100 REM CAP 4,PROB 15
110 DATA 92,63,75,82,72,53,100,89,70,81
120 READ A,B,C,D,E,F,G,H,I,J
130 PRINT (A+B+C+D+E+F+G+H+I+J)/10
140 END
```

17.
```
100 REM CAP 4,PROB 17
110 PRINT "TASSO DI INTERESSE FISSATO (%) "
120 INPUT R
130 PRINT "QUANTE VOLTE E' COMPOSTO ALL'ANNO "
140 INPUT M
150 LET T=((1+R/(100*M))↑M-1)*100
160 PRINT "IL TASSO REALE ANNUO DI INTERESSE E'"
170 PRINT T
180 END
```

19.
```
100 REM CAP 4,PROB 19
110 PRINT "INVESTIMENTO INIZIALE ";
120 INPUT P
130 PRINT "TASSO DI INTERESSE ANNUO (%) ";
140 INPUT I
150 PRINT "ANNI DI ACCUMULO INTERESSI ";
160 INPUT N
170 LET T=P*(1+I/100)↑N
180 PRINT "IL VALORE TOTALE E' ";T
190 END
```

CAPITOLO 5

1.
```
100 REM CAP 5,PROB 1
110 INPUT X,Y
120 IF X>Y THEN 150
130 PRINT Y
140 GOTO 160
150 PRINT X
160 END
```

3.
```
100 REM CAP 5,PROB 3
110 LET S=0
120 LET X=1
130 LET S=S+X
140 LET X=X+1
150 IF X <= 100 THEN 130
160 PRINT S
170 END
```

5. OUT OF DATA ERROR IN LINE 120

7.
```
100 REM CAP 5,PROB 7
110 LET S=0
120 READ X
130 IF X=9999 THEN 180
140 IF X<-10 THEN 120
150 IF X>10 THEN 120
160 LET S=S+X
170 GOTO 120
180 PRINT S
190 DATA -1,22,17,-6,4,7,9999
200 END
```

9.
```
100 REM CAP 5,PROB 9
110 LET C=1
120 LET T=0
```

```
130 LET W=10000
140 LET T=T+W
150 LET C=C+1
160 LET W=2*W
170 IF C<= 22 THEN 140
180 PRINT T
190 END
```

11. Sarà stampato il numero 83. Il programma trova il maggiore dei numeri contenuti nelle due istruzioni DATA.

13.
```
100 REM CAP 5,PROB 13
110 PRINT "PREZZO DI LISTINO (L) ";
120 INPUT L
130 PRINT "TASSO DI SCONTO (%) ";
140 INPUT R
150 LET D=L*(1 - R/100)
160 PRINT "IL PREZZO SCONTATO E'"
170 PRINT D;" LIRE"
180 END
```

15.
```
100 REM CAP 5,PROB 15
110 INPUT A,B
120 IF A >= 10 THEN 140
130 GOTO 180
140 IF B >= 10 THEN 160
150 GOTO 180
160 PRINT A+B
170 GOTO 280
180 IF A < 10 THEN 200
190 GOTO 240
200 IF B < 10 THEN 220
210 GOTO 240
220 PRINT A*B
230 GOTO 280
240 IF A < B THEN 270
250 PRINT A-B
260 GOTO 280
270 PRINT B-A
280 END
```

17.
```
100 REM CAP 5,PROB 17
110 PRINT "TASSO DI CRESCITA (%) ";
120 INPUT R
130 LET N=0
140 LET Q=1
150 LET Q=Q*(1+R/100)
160 LET N=N+1
170 IF Q <= 2 THEN 150
```

```
180 PRINT "IL NUMERO DEI PERIODI DI CRESCITA
    PER IL RADDOPPIO E' ";N
190 END
```

CAPITOLO 6

1.
```
100 REM CAP 6,PROB 1
110 PRINT " N"," SQR(N)"
120 PRINT
130 FOR N=2 TO 4 STEP .2
140 PRINT N,SQR(N)
150 NEXT N
160 END
```

3.
```
100 REM CAP 6,PROB 3
110 INPUT N
120 FOR X=2 TO N STEP 2
130 PRINT X
140 NEXT X
150 END
```

5.
```
         ABCDEFGHIJ
        ABCDEFGHIJ
       ABCDEFGHIJ
      ABCDEFGHIJ
     ABCDEFGHIJ
    ABCDEFGHIJ
   ABCDEFGHIJ
  ABCDEFGHIJ
 ABCDEFGHIJ
ABCDEFGHIJ
```

7. Siccome le iterazioni Z e V sono incrociate, comparirà un messaggio di errore.

9. Legge e stampa cinque numeri arrotondati a due decimali.

11. Visualizza 1.

13.
```
100 REM CAP 6,PROB 13
110 INPUT N
120 INPUT X
130 LET L=X
140 LET H=X
150 LET S=X
160 FOR I=1 TO N-1
```

```
170 INPUT X
180 IF X > L THEN 200
190 LET L=X
200 IF X < H THEN 220
210 LET H=X
220 LET S=S+X
230 NEXT I
240 PRINT "IL VOTO PIU' ALTO E' ";H
250 PRINT "IL VOTO PIU' BASSO E' ";L
260 PRINT "LA MEDIA E' ";S/N
270 END
```

15.
```
1 2 3
2 4 6
3 6 9
4 8 12
```

17.
```
100 REM CAP 6,PROB 17
110 READ N
120 FOR I=1 TO N
130 READ M,R,D1,D2,D3,D4,D5
140 PRINT "MATRICOLA ";M
150 LET H=D1+D2+D3+D4+D5
160 IF H <= 40 THEN 190
170 LET P=R*40 + 1.5*R*(H-40)
180 GOTO 200
190 LET P=R*H
200 PRINT "LA PAGA E' ";P
210 NEXT I
220 DATA 5
230 DATA 2,4800,8,10,8,7,10
240 DATA 5,3750,7,8,8,6,10
250 DATA 1,3250,8,10,6,8,8
260 DATA 4,5000,8,10,6,10,6
270 DATA 3,4250,6,6,8,10,7
280 END
```

CAPITOLO 7

1.
```
100 REM CAP 7,PROB 1
110 DIM X(20)
120 READ N
130 FOR I=1 TO N
140 READ X(I)
150 NEXT I
160 FOR I=1 TO N
170 PRINT X(I)
180 NEXT I
```

```
190 DATA 12
200 DATA 2,1,4,3,2,4,5,6,3,5,4,1
210 END
```

3.
```
100 REM CAP 7,PROB 3
110 DIM A(10,10)
120 INPUT N
130 FOR R=1 TO N
140 FOR C=1 TO N
150 INPUT A(R,C)
160 NEXT C
170 NEXT R
180 LET S=0
190 FOR I=1 TO N
200 LET S=S + A(I,I)
210 NEXT I
220 PRINT "LA SOMMA SULLA DIAGONALE PRINCIPALE E' ";S
230 END
```

5.
```
100 REM CAP 7,PROB 5
110 DIM A(15,15)
120 INPUT M,N
130 FOR R=1 TO M
140 FOR C=1 TO N
150 INPUT A(R,C)
160 NEXT C
170 NEXT R
180 LET S=0
190 FOR R=1 TO M
200 FOR C=1 TO N
210 LET S=S+A(R,C)
220 NEXT C
230 NEXT R
240 PRINT "LA SOMMA DEGLI ELEMENTI E' ";S
250 END
```

7. 10

9. 16

11.
```
100 REM CAP 7,PROB 11
110 DIM X(100)
120 INPUT N
130 FOR I=1 TO N
140 INPUT X(I)
150 NEXT I
160 FOR I=1 TO N-1
170 IF X(I) >= X(I+1) THEN 220
180 LET T=X(I+1)
```

```
190 LET X(I+1)=X(I)
200 LET X(I)=T
210 GOTO 160
220 NEXT I
230 FOR I=1 TO N
240 PRINT X(I);
250 NEXT I
260 END
```

13.
```
1 1 1 1 1 1
0 0 0 0 0 0
0 0 1 1 1 1
0 0 0 0 0 0
0 0 0 0 1 1
0 0 0 0 0 0
```

15.
```
100 REM CAP 7,PROB 15
110 DIM X(2,5)
120 FOR R=1 TO 2
130 FOR C=1 TO 5
140 READ X(R,C)
150 NEXT C
160 NEXT R
170 DATA 2,1,0,5,1
180 DATA 3,2,1,3,1
190 FOR R=1 TO 2
200 FOR C=1 TO 5
210 PRINT X(R,C);
220 NEXT C
230 PRINT
240 NEXT R
250 END
```

17.
```
100 REM CAP 7,PROB 17
110 DIM X(20,20)
120 INPUT M,N
130 FOR R=1 TO M
140 FOR C=1 TO N
150 INPUT X(R,C)
160 NEXT C
170 NEXT R
180 FOR R=1 TO M
190 LET S=0
200 FOR C=1 TO N
210 LET S=S+X(R,C)
220 NEXT C
230 PRINT "LA SOMMA DELLA RIGA ";R;" E' ";S
240 NEXT R
250 FOR C=1 TO N
260 LET P=1
```

```
270 FOR R=1 TO M
280 LET P=P*X(R,C)
290 NEXT R
300 PRINT "IL PRODOTTO DELLA COLONNA ";C;" E' ";P
310 NEXT C
320 END
```

19.
```
100 REM CAP 7,PROB 19
110 DIM X(4,6)
120 FOR R=1 TO 4
130 FOR C=1 TO 6
140 READ X(R,C)
150 NEXT C
160 NEXT R
170 DATA 48,40,73,120,100,90
180 DATA 75,130,90,140,110,85
190 DATA 50,72,140,125,106,92
200 DATA 108,75,92,152,91,87
210 FOR C=1 TO 6
220 LET S=0
230 FOR R=1 TO 4
240 LET S=S+X(R,C)
250 NEXT R
260 PRINT "IL TOTALE VENDITE GIORNALIERE ";C;" E' ";S
270 NEXT C
280 PRINT
290 FOR R=1 TO 4
300 LET S=0
310 FOR C=1 TO 6
320 LET S=S+X(R,C)
330 NEXT C
340 PRINT "IL TOTALE VENDITE PER AGENTE ";R;" E' ";S
350 NEXT R
360 LET S=0
370 FOR R=1 TO 4
380 FOR C=1 TO 6
390 LET S=S+X(R,C)
400 NEXT C
410 NEXT R
420 PRINT
430 PRINT "IL TOTALE VENDITE SETTIMANALI E'";S
440 END
```

21.
```
100 REM CAP 7,PROB 21
110 DIM P(20),X(20)
120 PRINT "LUNGHEZZA DEGLI ARRAY ";
130 INPUT N
140 FOR I=1 TO N
150 LET P(I)=I
160 NEXT I
170 FOR R=1 TO N
```

```
180 INPUT X(R)
190 NEXT R
200 FOR I=1 TO N-1
210 IF X(P(I))>X(P(I+1)) THEN 260
220 LET T=P(I)
230 LET P(I)=P(I+1)
240 LET P(I+1)=T
250 GOTO 200
260 NEXT I
270 PRINT " P"," X"
280 PRINT
290 FOR I=1 TO N
300 PRINT P(I),X(I),X(P(I))
310 NEXT I
320 END
```

CAPITOLO 8

1.
```
100 REM CAP 8,PROB 1
110 INPUT A$
120 FOR I=1 TO LEN(A$)
130 PRINT MID$(A$,I,1)
140 NEXT I
150 END
```

3.
```
100 REM CAP 8,PROB 3
110 INPUT A$
120 LET A=0
130 LET E=0
140 LET I=0
150 LET O=0
160 LET U=0
170 FOR J=1 TO LEN(A$)
180 IF MID$(A$,J,1) = "A" THEN 240
190 IF MID$(A$,J,1) = "E" THEN 260
200 IF MID$(A$,J,1) = "I" THEN 280
210 IF MID$(A$,J,1) = "O" THEN 300
220 IF MID$(A$,J,1) = "U" THEN 320
230 GOTO 330
240 LET A=A+1
250 GOTO 330
260 LET E=E+1
270 GOTO 330
280 LET I=I+1
290 GOTO 330
300 LET O=O+1
310 GOTO 330
320 LET U=U+1
330 NEXT J
340 PRINT "A = ";A
```

```
350 PRINT "E = ";E
360 PRINT "I = ";I
370 PRINT "O = ";O
380 PRINT "U = ";U
390 END
```

5.
```
100 REM CAP 8,PROB 5
110 INPUT A$
120 FOR I=1 TO LEN(A$)
130 IF MID$(A$,I,1)=CHR$(32) THEN 150
140 LET B$=B$ + MID$(A$,I,1)
150 NEXT I
160 PRINT B$
170 END
```

7.
```
100 REM CAP 8,PROB 7
110 LET C=0
120 FOR K=1 TO 5
130 INPUT A$
140 IF MID$(A$,1,4)<>"IL " THEN 160
150 LET C=C+1
160 FOR I=2 TO LEN(A$)-3
170 IF MID$(A$,I,4)<>"IL " THEN 190
180 LET C=C+1
190 NEXT I
200 NEXT K
210 PRINT C
220 END
```

9.
```
100 REM CAP 8,PROB 9
110 INPUT A$
120 LET C=0
130 FOR K=1 TO LEN(A$)-1
140 IF MID$(A$,K,2)<>"IN" THEN 160
150 LET C=C+1
160 NEXT K
170 PRINT C
180 END
```

11. Cancellate le righe dalla 190 alla 250. Aggiungete le righe 185 e 355.

```
185 FOR R=1 TO 3
355 NEXT R
```

CAPITOLO 9

1.
```
25 5 20
65
```

3.
```
100 REM CAP 9,PROB 3
110 DEF FNA(R)=3.14159*R↑2
120 DEF FNB(R)=4*3.14159*R↑3/3
130 PRINT
140 PRINT " R","AREA DEL" , "VOLUME DELLA"
150 PRINT TAB(10);"CERCHIO" , "SFERA"
160 PRINT
170 FOR R=1 TO 5 STEP .5
180 PRINT R;TAB(7);FNA(R),FNB(R)
190 NEXT R
200 END
```

5.
```
55
15
36
```

7.
```
500 REM SUBROUTINE
510 LET L=X(2)
520 FOR I=3 TO X(1)+1
530 IF L >= X(I) THEN 550
540 LET L=X(I)
550 NEXT I
560 RETURN
```

9.
```
900 REM SUBROUTINE
910 LET S1=0
920 LET S2=0
930 FOR I=2 TO Y(1)+1
940 LET S1=S1+Y(I)
950 LET S2=S2+Y(I)↑2
960 NEXT I
970 LET M=S1/Y(1)
980 LET S=SQR((Y(1)*S2-S1↑2)/(Y(1)*(Y(1)-1)))
990 RETURN
```

CAPITOLO 10

1. Sarà stampato S OK.

3. Modificate la riga 1010 nel modo seguente:

```
1010 IF R<1 OR R>25 OR C<1 OR C>40 THEN
PRINT "FUORI DALLO SCHERMO":END
```

5. Una modifica dell'esempio 4 dà:

```
90 PRINT CHR$(28);
100 PRINT CHR$(147);
110 LET J=1
120 PRINT CHR$(19);
125 LET N=N + 1
130 FOR I=1 TO N
140 PRINT
150 NEXT I
160 PRINT TAB(J);"┌──┐"
170 PRINT TAB(J);"│  │"
180 PRINT TAB(J);"│  │"
190 PRINT TAB(J);"└──┘"
200 LET J=J + 1
210 IF J > 33 THEN END
215 IF I > 20 THEN END
220 PRINT CHR$(147);
230 GOTO 120
240 END
```

CAPITOLO 11

1.
```
50 REM CAP 11,PROB 1
100 FOR I=1 TO 25
110 LET N=INT(100*RND(1))/10
120 PRINT N,
130 NEXT I
140 END
```

3. Una tipica visualizzazione è:

```
.04 .02 .17 .14
.01 .03 .16 .04
.05 .17 .19 .12
.07 .11 .19 .11
.19 .2  .18 .04
```

5.
```
50 REM CAP 11,PROB 5
100 FOR I=1 TO 5
110 READ N
120 LET H=0
130 LET T=0
140 FOR J=1 TO N
```

```
150 LET X=INT(2*RND(1)+1)
160 IF X=1 THEN 190
170 LET T=T+1
180 GOTO 200
190 LET H=H+1
200 NEXT J
210 PRINT
220 PRINT "PER ";N;" LANCI CI SONO"
230 PRINT H;" TESTA E ";T;" CROCE"
240 NEXT I
250 DATA 10,50,100,500,1000
260 END
```

7.
```
50 REM CAP 11,PROB 7
100 LET S=0
110 FOR I=1 TO 100
120 LET S=S+RND(1)
130 NEXT I
140 PRINT S/100
150 END
```

9.
```
50 REM CAP 11,PROB 9
100 LET M=0
110 FOR I=1 TO 1000
120 LET A=60*RND(1)
130 LET B=60*RND(1)
140 IF ABS(A-B)>10 THEN 160
150 LET M=M+1
160 NEXT I
170 PRINT "LA PROBABILITA' DI INCONTRARSI E' ";M/1000
180 END
```

11.
```
50 REM CAP 11,PROB 11
100 FOR I=1 TO 25
110 LET S=0
120 FOR J=1 TO 12
130 LET S=S+RND(1)
140 NEXT J
150 LET R=10+2*(S-6)
160 PRINT INT(100*R+.5)/100
170 NEXT I
180 END
```

13.
```
50 REM CAP 11,PROB 13
100 DIM A(11)
110 FOR N=1 TO 1000
120 LET SOMMA=INT(6*RND(1)+1)+INT(6*RND(1)+1)
130 FOR I=2 TO 12
140 IF SOMMA=I THEN LET A(I-1)=A(I-1) + 1:GOTO 160
```

```
150 NEXT I
160 NEXT N
170 FOR I=2 TO 12
180 PRINT I;A(I-1)
190 NEXT I
200 END
```

CAPITOLO 12

1. Una possibile struttura del record potrebbe essere:

Variabile	Descrizione	Lunghezza approssimativa
TITLC$	Titolo della cassetta	50
MARCA$	Marca della cassetta	20
NOME$	Nome dell'autore	50
TIPO$	Tipo di musica	15
PREZZO$	Prezzo della cassetta	5

Il record dovrebbe essere lungo 145 dove 5 spazi extra sono stati aggiunti come spazi separatori.

3. Una possibile struttura del record è:

Variabile	Descrizione	Lunghezza approssimativa
NOME$	Nome dell'articolo	50
POS$	Posizione dell'articolo	15
AMM	Valore dell'articolo	10

La lunghezza del record dovrebbe essere 78 con 3 spazi aggiunti per separare gli elementi.

5. Il programma per calcolare gli importi mensili e il totale degli importi addebitati su ciascuna carta può essere il seguente:

```
100 REM CAP 12,PROB 5
110 LET I=1
120 PRINT "CARTA DI CREDITO: ";
130 INPUT CART$
140 IF CART$="FINE" THEN 320
150 PRINT "NOME DEL NEGOZIO: ";
160 INPUT NOME$
170 PRINT "DATA DELL'ACQUISTO: ";
180 INPUT DATE$
190 PRINT "DESCRIZIONE DELL'ACQUISTO: ";
```

```
200 INPUT DES$
210 PRINT "IMPORTO SPESO: ";
220 INPUT IMP
230 OPEN 15,8,15
240 OPEN 2,8,2,"CHARGE,L," + CHR$(125)
250 PRINT#15,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
260 GOSUB 1000
270 PRINT#2,CART$;",";NOME$;",";DATE$;",";DES$;",";IMP
280 CLOSE 2
290 CLOSE 15
300 LET I=I + 1
310 GOTO 120
320 REM SOMMA GLI ADDEBITI
330 LET SUM=0
340 PRINT "CARTA DI CREDITO"
345 PRINT "DI CUI CALCOLARE IL SALDO: ";
350 INPUT TEMP$
360 IF TEMP$="FINE" THEN END
370 OPEN 15,8,15
380 OPEN 2,8,2,"CHARGE,L," + CHR$(125)
390 FOR K=1 TO I - 1
400 PRINT#15,"P"CHR$(2)CHR$(K)CHR$(0)CHR$(1)
410 GOSUB 1000
420 INPUT#2,CART$,NOME$,DATE$,DES$,IMP
430 IF CART$ <> TEMP$ THEN 450
440 LET SUM=SUM + IMP
450 NEXT K
460 PRINT "IL SALDO DELLA "; TEMP$;" E' ";SUM
470 CLOSE 2
480 CLOSE 15
490 GOTO 320
1000 REM VERIFICA LA PRESENZA DEL RECORD
1010 INPUT#15,ERRNUM,ERRNAME$
1020 IF ERRNUM=50 OR ERRNUM=0 THEN RETURN
1030 PRINT ERRNUM,ERRNAME$
1040 CLOSE 2
1050 CLOSE 15
1060 END
```

Nota: Questo programma non accumula le informazioni sul file. Potete fare in modo di aggiungere informazioni al file, come nell'Esempio 2.

Indice analitico

G

I

L

M

N

O

P

R

La McGraw-Hill pubblica in tutto il mondo centinaia di libri di informatica per lo studio, la professione e il tempo libero. La produzione in lingua italiana comprende:

88 7700 001 5	J. Heilborn e R. Talbott, *Guida al Commodore 64*
88 7700 002 3	C.A. Street, *La gestione delle informazioni con lo ZX Spectrum*
88 7700 003 1	T. Woods, *L'Assembler per lo ZX Spectrum*
88 7700 004 X	R. Jeffries, G. Fisher e B. Sawyer, *Divertirsi giocando con il Commodore 64*
88 7700 005 8	G. Bishop, *Progetti hardware con lo ZX Spectrum*
88 7700 006 6	H. Mullish e D. Kruger, *Il BASIC Applesoft*
88 7700 007 4	N. Williams, *Progettazione di giochi d'avventura con lo ZX Spectrum*
88 7700 008 2	H. Peckham, *Il BASIC e il PC-IBM in pratica*
88 7700 009 0	H. Peckham, *Il BASIC e il Commodore 64 in pratica*
88 7700 010 4	S. Nicholls, *Tecniche avanzate in Assembler con lo ZX Spectrum*
88 7700 012 0	S. Kamins e M. Waite, *Programmazione umanizzata in Applesoft*
88 7700 013 9	A. Pennell, *Guida allo ZX Microdrive e all'Interface 1*
88 7700 015 5	P. Cohen, *Grafica e animazione con gli Apple II*
88 7700 016 3	D. Duff, *Guida al Macintosh*
88 7700 017 1	G. Kane, *Il manuale MC68000*
88 7700 018 X	P. Hoffman e T. Nicoloff, *Il manuale MS-DOS*

In preparazione

88 7700 011 2	K. Skier, *L'Assembler per il VIC 20 e il Commodore 64*
88 7700 020 1	S. Nicholls, *Grafica avanzata con lo ZX Spectrum*
88 7700 021 X	L. Graham e T. Field, *Guida al PC-IBM*
88 7700 022 8	T. Field, *Come usare MacWrite e MacPaint*

Il successo che i libri di Herbert Peckham hanno incontrato in tutto il mondo è dovuto al metodo pratico con cui affrontano l'argomento della programmazione in BASIC.
Questo in particolare è dedicato al C-64, uno dei più diffusi home computer.
Il metodo pratico di Peckham, l'Hands-on-BASIC, accompagna gradualmente l'utente, al quale non è richiesta alcuna conoscenza matematica o informatica di base, dai primi approcci alla tastiera fino alla completa padronanza del computer e della programmazione.
Ogni capitolo, dedicato ad uno o più argomenti, è così suddiviso:
– una fase di "scoperta" nella quale il lettore è invitato a provare alcune istruzioni e a capirne l'effetto;
– una fase teorica di consolidamento dove i concetti vengono dettagliatamente spiegati;
– una serie di esercizi di revisione ai quali il lettore deve rispondere e attraverso i quali può valutare il proprio grado di apprendimento e decidere se passare al successivo capitolo o ritornare ad approfondire quanto gli è stato appena spiegato.
Queste caratteristiche fanno sì che **Il BASIC e il COMMODORE 64 in pratica** costituisca un formidabile strumento didattico adatto sia per un uso individuale che come testo nei corsi di informatica di base.
Durante la trattazione sono esaminati in dettaglio numerosi programmi completi immediatamente utilizzabili.

Un libro di BYTE

Lire 27 000
(IVA 2% inclusa)

ISBN 88 7700 009 0